UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

INTEGRAÇÃO, GERENCIAMENTO E IMPLEMENTAÇÃO DIDÁTICA DE CÉLULAS FLEXÍVEIS DE MANUFATURA

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA MECÂNICA

GUILHERME ERNANI VIEIRA

FLORIANÓPOLIS, JUNHO - 1996.

INTEGRAÇÃO, GERENCIAMENTO E IMPLEMENTAÇÃO DIDÁTICA DE CÉLULAS FLEXÍVEIS DE MANUFATURA

GUILHERME ERNANI VIEIRA

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA MECÂNICA, ÁREA DE CONCENTRAÇÃO FABRICAÇÃO, APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA DA UNIVERSIDADE FEDERAL DE SANTA CATARINA.

João Carlos Espíndola Ferreira

Prof. João Carlos Espíndola Ferreira, Ph.D. - Orientador

Prof. Abelardo Alves de Queiroz - Coordenador do Curso

BANCA EXAMINADORA:

Prof. Dr.Ing. Carlos Alberto Martin - Presidente

Prof. Abelardo Alves de Queiroz, Ph.D.

Prof. Gregório Varvakis Rados, Ph.D.

Eng°. Manoel C. L. F. Braga, M.Sc.

À minha esposa Karla e

à vó Dorça, que nos deu tantas alegrias.

AGRADECIMENTOS

A Deus, porque sem Ele este trabalho não poderia ter sido feito.

À Karla, que com muita paciência e carinho me apoiou durante todo trabalho.

Aos meus pais, pessoas simplesmente incríveis.

Ao meu orientador, por toda atenção e dedicação recebida.

Ao meu amigo Carlão pelas nossas discussões e cervejas.

Ao prof. Jean-Marie Farines, por permitir a utilização do Laboratório de Automação Industrial.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela bolsa de estudos.

Ao GRUCON, que sempre me deu todo apoio e recursos que precisei.

Ao Departamento de Engenharia Mecânica, pela acolhida recebida.

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO 2:

CAPÍTULO 3:

CAPÍTULO 4:

CAPÍTULO 5:

LISTA DE SIGLAS

AGV	Automated Guided Vehicle	Veículo Guiado Automaticamente
AI	Artificial Inteligence	Intelegência Artificial
AS/RS	Automated Storage and Retrieval System	Sistema Automatizado de Armazenamento e Retirada de Materiais
CAD	Computer Aided Design	Projeto Auxiliado por Computador
CAM	Computer Aided Manufacturing	Manufatura Auxiliada por Computador
CAPP	Computer Aided Process Planning	Planejamento do Processo Auxiliado por Computador
CCC		Computador Central da Célula
CIM	Computer Integrated Manufacturing	Manufatura Integrada por Computador
CNC	Computer Numerical Control	Comando Numérico Computadorizado
DC	Direct Current	Corrente Contínua
ES	Engineering System	Sistema de Engenharia
EM		Estação de Montagem
FAC		Funções de Atendimento à Célula
FAU		Funções de Atendimento ao Usuário
FMC	Flexible Manufacturing Cell	Célula Flexível de Manufatura
FMS	Flexible Manufacturing System	Sistema Flexível de Manufatura
GT	Group Technology	Tecnologia de Grupo
JA		Janela de Apresentação
JIT	Just-In-Time	Just-In-Time
LAI		Laboratório de Automação Industrial
LAN	Local Area Network	Rede de Comunicação Local
LED	Light Emissor Diode	Diodo Emissor de Luz
M		Menu
MAP	Manufacturing Automation Protocol	Protocolo de Automação da Manufatura
MG		Mesa Giratória

MRP	Material Requirements Planning	Planejamento dos Requisitos de Material
NN	Neural Networks	Redes Neurais
OOP	Object-Oriented Programming	Programação Orientada por Objetos
PB		Peça Bruta
PC	Personal Computer	Computador Pessoal
PCS	Production Control System	Sistema de Controle da Produção
PH		Programa Hospedeiro
RdP		Rede de Petri
RPI		Rede de Petri Interpretada
SV		Sistema de Visão
TOP	Technical and Office Protocol	Protocolo Técnico e de Escritório
TQC	Total Quality Control	Controle da Qualidade Total
VE		Variável Externa
VM		Velocidade Máxima
WAN	Wide Area Network	Rede de Comunicação de Longa Distância
WIP	Work-In-Process	Estoque em Processo (ou Intermediário)

RESUMO

Este trabalho tem como objetivo mostrar o desenvolvimento de um sistema produtivo automatizado formado pela integração de duas células flexíveis de manufatura que simula a fabricação e a montagem de peças. Cada FMC foi desenvolvida a partir da utilização de dispositivos didáticos, tais como: robôs XR4 e SCARA, esteiras transportadoras de peças e estações de montagem. A integração desses dispositivos numa célula foi feita através do controlador do robô, o qual é comandado pelo *software* de gerenciamento da célula (gerente FMC) através do *driver* de comunicação desenvolvido neste trabalho. As características, atribuições, estrutura de execução interna e estrutura dos menus dos gerentes FMC estão detalhadamente descritos. É feita também uma descrição de redes de Petri interpretadas, as quais são utilizadas nos gerentes FMC para implementação das atividades de atendimento à célula, como por exemplo: controlar os robôs, coordenar a sequência das atividades a serem executadas pela célula e estabelecer a integração e o sincronismo entre as FMCs. A integração das células é feita através de duas esteiras transportadoras, a partir da troca de informações entre os gerentes das células.

ABSTRACT

This work aims at showing the development of an automated manufacturing system implemented through the integration of two flexible manufacturing cells that simulates the machining and assembly of parts. Each FMC was developed with didactic devices such as: XR4 and SCARA robots, conveyors for part transportation and assembly stations. The integration of these components in a cell was made through the robot controller, which is managed by the cell management software (FMC manager) via the communication driver software. The characteristics, functions, internal execution structure and menu structure of the FMC managers are described in detail. It is also given a description of Petri nets, which are used by the FMC managers to implement the cell activities such as: robot control, coordination of the sequence of the activities to be performed in the cell and the integration and sincronism between the FMCs. The integration of the cells is performed by two conveyors, together with a flow of information between the cell managers.

1

INTRODUÇÃO

Neste trabalho, desenvolveu-se um sistema produtivo automatizado que tem como objetivo simular a fabricação e a montagem de peças.

Esse sistema foi desenvolvido a partir da integração de duas células flexíveis de manufatura (FMCs) formadas por componentes didáticos, a saber: máquinas, esteiras, robôs e estações de montagem.

Essas células são controladas ou gerenciadas por *sofwares* dedicados, isto é, cada FMC tem um programa responsável pelo gerenciamento das atividades a serem realizadas pela célula (gerente FMC).

As funções de gerenciamento das atividades das células, como por exemplo, a movimentação dos robôs, esteiras e estações de montagem, leitura dos sinais de entrada e atuação através dos sinais de saída, etc, são modeladas por redes de Petri interpretadas (RPIs).

Existem também funções de atendimento ao usuário (ou operador) da célula, como: iniciar ou abortar atividades, acompanhar o desempenho, verificar *status* dos dispositivos, etc, as quais deverão ser acessadas através da interface gráfica dos gerentes das células.

A integração física (a nível de *hardware*) dos componentes de uma FMC foi feita por meio do controlador do robô (MARK-IV), o qual é conectado serialmente ao computador central da célula.

Os gerentes FMCs devem poder trocar informações para estabelecer a integração lógica das FMCs. Nos gerentes, essa integração também é especificada através das RPIs das células.

Pretende-se também fazer uma aplicação prática do sistema produtivo proposto, através da fabricação e montagem de quatro tipos de peças.

Alguns fatores que motivaram o desenvolvimento deste trabalho foram, por exemplo:

- a possibilidade de se desenvolver *softwares* com aplicabilidade prática;
- a disponibilidade de equipamentos;
- por ser esta uma área onde a pesquisa ainda está num estágio inicial;
- o trabalho poderá auxiliar determinadas disciplinas ministradas na UFSC, principalmente as relacionadas à área de automação industrial;
- novos tipos de trabalhos poderão ser desenvolvidos a partir dos fundamentos deixados por este; e
- o sistema produtivo proposto nunca foi implementado na UFSC.

Tanto em instituições acadêmicas como em indústrias, centenas de profissionais estão atuando nas áreas de manufatura integrada por computador, sistemas flexíveis de manufatura e células flexíveis de manufatura. É fácil perceber que cada um deles tem suas próprias definições e modelos para CIM, FMS e FMC. Por esta razão, procura-se mostrar no capítulo 2, que é o de revisão da literatura, um pouco sobre essas diversidades de conceitos e definições. Neste capítulo também deverão ser apresentados conceitos do autor, haja visto que, pela variedade de idéias existentes, ainda não se tem definições que possam ser consideradas "padrões". Deverão ser tratados também assuntos relativos ao desenvolvimento de *softwares* com aplicação em sistemas de manufatura, como por exemplo, o integrador CIM, o gerente FMS e o gerente FMC. Algumas ferramentas que estão

sendo aplicadas nesses sistemas, como inteligência artificial, redes neurais, redes de Petri e programação orientada por objetos, junto com uma rápida descrição sobre redes e protocolos de comunicação, deverão também ser abordados naquele capítulo.

No terceiro capítulo é feita uma descrição geral do trabalho prosposto, detalhando-se os objetivos desejados e as ferramentas a serem utilizadas na integração e gerenciamento em ambas as células flexíveis de manufatura. Dentre essas ferramentas, cita-se a utilização de redes de Petri interpretadas e a interface gráfica INWIN.

O controlador MARK-IV, os robôs XR4 e SCARA, junto com os *drivers* de comunicação a serem desenvolvidos, serão detalhadamente descritos no quarto capítulo.

No quinto e sexto capítulos, deverão ser abordados a integração e o desenvolvimento dos *softwares* de gerenciamento da FMC1 e da FMC2, respectivamente.

No capítulo 7 deseja-se mostrar como será feita a integração física e lógica das FMCs, e no capítulo 8 é feita uma análise final de avaliação do trabalho desenvolvido.

2

REVISÃO DA LITERATURA

O que se entende por manufatura integrada por computador? Sistemas flexiveis de manufatura? Células flexiveis de manufatura? Existem dezenas de definições e modelos de CIM/FMS/FMC sendo aplicados nas diversas partes do mundo, cada qual adequado ou voltado aos seus interesses, às suas configurações. Alguns deles puramente teóricos (ou didáticos), enquanto outros são resultantes da prática do dia-a-dia nas indústrias.

Este capítulo procura mostrar um pouco sobre essa diversidade de conceitos, junto com alguns modelos, metodologias de implementação, etapas de desenvolvimento e requisitos.

Os *softwares* que gerenciam os sistemas CIM/FMS/FMC serão descritos, salientando-se suas principais atribuições e características.

A palavra "integração" num sistema CIM não poderia de forma alguma existir sem a presença das redes de comunicação. Por esta razão, as redes e alguns protocolos de comunicação também serão descritos.

Resumidamente também serão citados alguns trabalhos nas áreas CIM, FMS e FMC que utilizam modernas filosofias computacionais como: inteligência artificial, redes de Petri e redes neurais. A programação orientada por objetos, ferramenta que começa a ser bastante aplicada em sistemas de manufatura, também será abordada.

2.1 - MANUFATURA INTEGRADA POR COMPUTADOR

Muitos profissionais definem CIM das mais variadas maneiras. Rembold et al. (1993) cita, por exemplo, cinco modelos/conceitos de CIM adotados por cinco entidades mundialmente conhecidas:

- o conceito de CIM para a IBM;
- o modelo hierárquico NIST[1]-AMRF[2];
- o conceito de CIM para a Siemens;
- o conceito de CIM para a DEC[3];
- o modelo ESPRIT[4]-CIM-OSA[5].

Para a *Intel Corporation*, diz James Kellso, CIM é "a interação entre pessoas e máquinas através do computador e da tecnologia de informação para integrar e, automaticamente, executar desenvolvimentos e tarefas de manufatura" [Kellso 1989].

"Enquanto nenhuma definição foi ainda genericamente aceita", diz Leslie King, "uma idéia, implícita na maioria das definições, é que CIM é toda atividade física, organizacional e gerencial, relacionada ao projeto, produção e distribuição de mercadorias produzidas por uma empresa de manufatura, incluindo funções internas e externas (p.ex., vendedores, compradores e subcontratações) com o resultado sendo realimentado (em malha-fechada), com o planejamento da manufatura e o sistema de controle integrados" [King 1991].

---

1 National Institute of Standards and Technology

2 Advanced Manufacturing Research Facility

3 Digital Equipment Corporation

4 European Strategic Programme for Research and Development in Information Technology

5 Open System Architecture

A *Computer and Automation Systems Association* (CASA), da Sociedade dos Engenheiros de Manufatura (SME), define CIM como sendo a "integração de toda a estrutura responsável pela manufatura através do uso de sistemas integrados e comunicações de dados, acoplados com novas filosofias gerenciais que melhoram a eficiência organizacional e pessoal" [Rehg 1994].

Nyman (1992) considera CIM como uma evolução do FMS, que por sua vez, evoluiu da FMC. Evoluir, neste conceito, refere-se ao aumento do nível de integração.

Com base nessas e também em algumas outras definições que não foram citadas aqui, CIM será definido neste trabalho como:

> *CIM é a integração de todas as atividades envolvidas na manufatura (p.ex., vendas, compras, projeto, planejamento, administração, finanças e produção), através de uma rede de comunicação e de um software de gerenciamento (integrador CIM), com o objetivo de melhorar a eficiência organizacional, pessoal e de produção.*

Como se pode observar, muitas áreas diferentes estão inseridas nesta definição. No presente trabalho, entretanto, apenas o setor da produção será estudado, entendido como sendo executado por um *sistema flexível de manufatura* (figura 2.1).

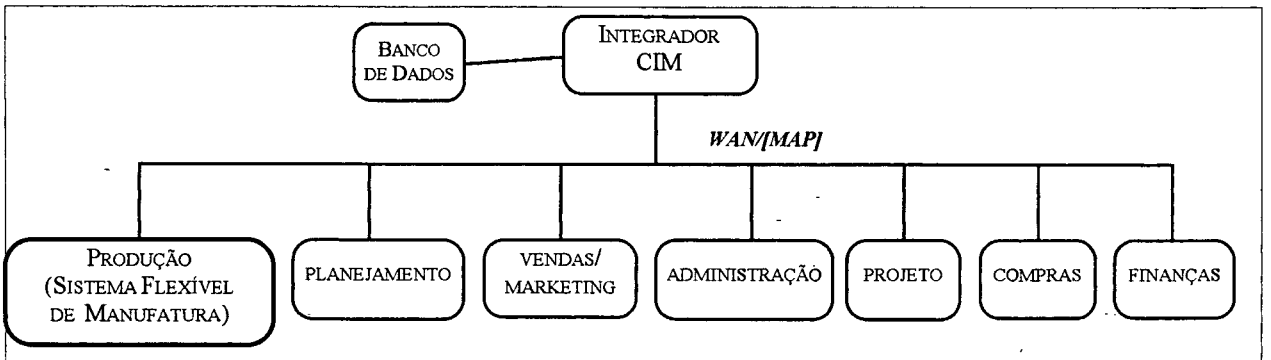


Figura 2.1 - Arquitetura CIM

A integração num sistema CIM é estabelecida fisicamente por uma rede de comunicação de longa distância *(WAN)*, ou, como também encontrado em algumas indústrias, por meio de uma rede de comunicação local (LAN), conectando-se os módulos (ou departamentos) com o computador central (integrador CIM ou computador corporativo), considerado o cérebro da integração da manufatura.

2.1.1 - O INTEGRADOR CIM

Uma definição para um integrador CIM diz que ele é um coordenador de processos de manufatura, informações e pessoas [Kaltwasser 1990]. Esse mesmo autor diz que em geral o melhor integrador para uma companhia é aquele que pode conciliar a maioria dos requisitos do sistema, sendo as necessidades específicas da planta que norteiam a definição de qual o melhor sistema a ser adquirido.

Pode-se dizer mais detalhadamente que o integrador CIM é responsável pelo gerenciamento da execução, supervisão e controle das atividades nos diversos setores da empresa, permitindo o interfaceamento entre esses setores de maneira que eles possam estar perfeitamente integrados.

Para Rembold et al. (1993), a eficiência de operação de uma planta CIM depende da qualidade e integridade de um "bem-projetado" sistema de gerenciamento das informações. Eles dizem ainda que esse sistema é "o coração da manufatura integrada por computador".

Algumas atividades que devem ser consideradas (e armazenadas) em um sistema de gerenciamento das informações para um sistema CIM são [Rembold et al. 1993]: previsão econômica, exigências de mercado, inovações de produto, desenvolvimento de materiais, dados de padronização e catálogos, desenvolvimento demográfico, comportamento do mercado, estratégias do

concorrente, atividades de vendas, atividades de engenharia, controle e planejamento da produção, compra de matéria-prima e equipamentos, finanças, políticas de inventário, atividades de vendas, etc.

Alguns requisitos gerais para um sistema CIM, que do ponto de vista deste trabalho podem ser também utilizadas para o desenvolvimento de um sistema FMS e/ou FMC (seções 2.2 e 2.3 respectivamente) são [Ohsen 1992]:

Integração dos Módulos - Todos os programas (módulos) devem ser completamente integrados, sem requerer entrada redundante de dados, isto é, entrada dos mesmos dados em diferentes módulos.

Crescimento do Sistema Futuro - O sistema deve estar preparado para a integração de novos módulos que possam vir a ser adquiridos no futuro.

Segurança do Sistema - O sistema deve ter algum tipo de segurança, como senhas (*passwords*), para garantir a segurança (ou sigilo) de certos tipos de informações, como por exemplo, o salário dos funcionários.

Treinamento - Um treinamento substancial não somente com módulos de programas mas também com os novos equipamentos.

Manutenção do Sistema - Tanto a parte de *software* quanto de *hardware* exigem uma manutenção contínua. Quando um sistema CIM é interrompido, a produção pode parar ou ficar severamente limitada. É importante que haja uma estratégia para responder rápido tanto aos problemas de *software* quanto de *hardware*.

Programação in-house - Embora a programação feita na própria companhia (*in-house*) deva ser evitada, é importante que parte do desenvolvimento seja feito (ou estruturado) pela própria companhia. Isto porque, se o fornecedor do sistema (ou módulo) sair do comércio ou interromper o suporte (manutenção), por exemplo, todo sistema poderá ficar comprometido.

2.1.2 - ALGUMAS CONSIDERAÇÕES ANTES DE SE IMPLEMENTAR CIM

De acordo com Aletan (1991), alguns pré-requisitos para a adoção da estratégia CIM são que "as empresas devem ter objetivos de negócios e de manufatura, devem conhecer seus clientes, competidores e os clientes dos competidores. Não é suficiente conhecer quem são os competidores, há também a necessidade de saber o quão bom eles são. Deve ser desenvolvido um plano de como a empresa está, onde ela quer estar ou para onde está indo. Junto com um plano de negócio, um plano de automação deve ser desenvolvido baseado nos objetivos de manufatura". Ele diz também que estratégias de automação devem ser planejadas junto com os objetivos de negócios e manufatura, entretanto, antes de se adotar essas estratégias, deve "haver um bom entendimento dos prós e contras da automação".

2.1.3 - METODOLOGIAS PARA IMPLEMENTAR CIM

Algumas metodologias, estratégias ou etapas desenvolvidas por diversos profissionais da área para a implementação da manufatura integrada por computador são descritas a seguir.

Para Aletan (1991) seis atividades de pré-implementação devem ser feitas antes de se adotar o CIM:

1°. Desenvolver o modelo da empresa - Deve haver um entendimento detalhado dos problemas da empresa.

2°. Desenvolver o modelo funcional - Objetivos de manufatura devem ser estabelecidos.

3°. Desenvolver um modelo informal - As interfaces do sistema, tipos de informações a serem compartilhadas, requisitos da base de dados e tecnologias disponíveis e aplicáveis devem ser estabelecidas.

4°. Desenvolver um modelo de rede - Devem ser estabelecidos os requisitos para as redes de comunicação que integrarão os sistemas.

5°. Desenvolver um modelo organizacional - Os efeitos e as implicações da integração dos módulos devem ser expostos. Os prós e contras da integração devem ser estabelecidos e analisados.

6°. Desenvolver um plano de implementação - O "como". "quando" e "o que" da integração devem ser definidos.

Uma metodologia para avaliar as oportunidades de se implementar CIM numa empresa, obtida de uma série de conceitos de sistemas de engenharia e de indústrias, foi proposto por Kaeli (1990), e está sumariamente descrito pelos seguintes passos:

1°. Monte uma equipe CIM e selecione um líder.

2°. Mobilize a equipe CIM.

a. Defina uma missão, um objetivo central.

3°. Identifique o modelo de como os negócios operam.

4°. Identifique as principais áreas funcionais (por exemplo, no departamento de vendas: previsões, no departamento de compras: preço, etc).

a. Defina uma matriz de participação, que relaciona departamentos e ações que deverão ser executadas (p.ex.,minimizar os gastos e desperdícios).

5°. Analise cada área funcional.

6°. Selecione os critérios relevantes para o CIM.

7°. Avalie e defina um *rank* das áreas funcionais.

8°. Completar uma matriz de oportunidades para o CIM, utilizada para identificar aquelas áreas funcionais que têm potenciais mais fortes para o CIM do que outras.

Para Kellso (1989), os blocos que suportam a implementação de qualquer CIM são:

- Os sistemas de comunicação e gerenciamento de informação.
- Os sistemas de controle e gerenciamento de materiais.
- Os sistemas de controle e gerenciamento da produção.
- A integração das informações e dos processos entre os vários elementos.

"Ninguém conhece as necessidades melhor do que aqueles envolvidos na operação do dia-a-dia, então ninguém pode planejar um sistema CIM melhor do que os próprios empregados da companhia", diz Ohsen (1992). Para ele, existem três passos básicos para se planejar um sistema CIM:

- Definir as necessidades do negócio - Para a empresa poder fazer uma decisão inteligente do quê comprar, se um sistema direcionado ao CIM ou um outro sistema qualquer de importância para a firma. Um levantamento formal das necessidades deve ser feito para desenvolver uma lista dos requisitos.
- Selecionar o *software* - Porque a funcionalidade do *software* é o que realmente satisfaz os requisitos de implantação do CIM.
- Selecionar o *hardware* - adequado para o *software* selecionado.

Para Rehg (1994), a implementação bem sucedida de um CIM segue um processo em três passos:

(1°) Conhecimento da empresa em três areas:

- Tecnologia - o nível atual da tecnologia e a sofisticação presente na manufatura.

- Recursos humanos - o estado atual da motivação dos empregados para a adoção do CIM na empresa.
- Sistemas - o *porquê* dos sistemas produtivos estarem operando como estão.

(2°) Simplificação - eliminação de desperdícios

Segundo Rehg, simplificação é o processo que remove desperdício de toda operação ou atividade para melhorar a produtividade e a eficiência do departamento e da organização. A implementação do CIM sem que haja a priori uma simplificação significa dizer que os desperdícios irão permanecer, podendo até ser aumentados após o CIM.

(3°) Implementação com medidas de desempenho

É a implementação de fato, que deve ser acompanhada periodicamente através do desempenho de alguns parâmetros, como:

- tempo de ciclo do produto;
- estoques;
- tempos de *setup* da produção;
- eficiência da manufatura;
- qualidade e retrabalho; e outros.

2.1.4 - BENEFÍCIOS DO CIM

Antes de se partir para a implementação de um sistema CIM é importante verificar se será conveniente para a empresa justificar o elevado investimento (de tempo, dinheiro, etc) a ser feito. Para Kaltwasser (1990), existem muitos fatores que justificam a implantação de um novo sistema ou a melhoria do sistema atual. Os mais óbvios são: redução nos custos de materiais e redução na mão-de-obra (direta e/ou indireta),

redução de refugos, retrabalhos e estoques. Os ítens menos óbvios, mas que também são muito importantes, são: redução no tempo de ciclo, aumento da capacidade, melhoramento na qualidade dos produtos e no serviço aos clientes, uma maior flexibilidade na produção e uma melhor posição no mercado.

2.1.5 - REDES DE COMUNICAÇÃO

A comunicação tem um papel fundamental na manufatura integrada por computador. A escolha de um sistema de comunicação fortemente determina a capacidade e produtividade de uma fábrica como um todo [Rembold et al. 1993].

Uma rede de comunicação é o dispositivo que integra fisicamente os componentes. Composta principalmente por *softwares*, estações repetidoras e meios (cabos, placas, etc), uma rede tem como objetivo permitir que componentes possam trocar informações, de maneira rápida e segura.

Existem diferentes tipos de topologias para redes de comunicação que estão sendo aplicadas em sistema de manufatura, sendo que as mais facilmente encontradas são do tipo estrela, anel, barramento e árvore (figura 2.2).

Também existem diversos procedimentos de acesso à rede, como: *polling, time-division multiplexing, carrier sense multiple access, token passing*, etc, entretanto eles não serão aqui abordados.

Se o leitor desejar, pode recorrer a [Rehg 1994] ou a [Rembold et al. 1993] para maiores informações sobre procedimentos de acesso, redes e protocolos de comunicação.

Talvez o maior problema na integração de sistemas de manufatura seja a necessidade de se interconectar dispositivos heterogêneos, quase sempre produzidos por fabricantes diferentes.

Uma solução que se tem buscado para tentar solucionar este problema é a padronização. No entanto, esta não tem sido

uma tarefa muito fácil pois cada usuário (uma indústria, por exemplo) tem em suas instalações configurações próprias e não está disposto a jogar fora todo o investimento feito. Um outro problema diz respeito à existência de inúmeros fabricantes de redes e protocolos. Cada qual não quer abdicar-se do seu produto.

Uma solução para esse impasse tem surgido através do modelo de referência OSI (*Open Systems Interconnection*) da ISO (*International Standards Organization*). O OSI é um modelo que pode ser utilizado como referência na implementação de arquiteturas de redes de comunicação orientadas a qualquer tipo de aplicação [Braga 1993]. Uma aplicação desse padrão está sendo feito para implementar as redes de comunicação de longa distância (WANs) e as redes de comunicação local (LANs).

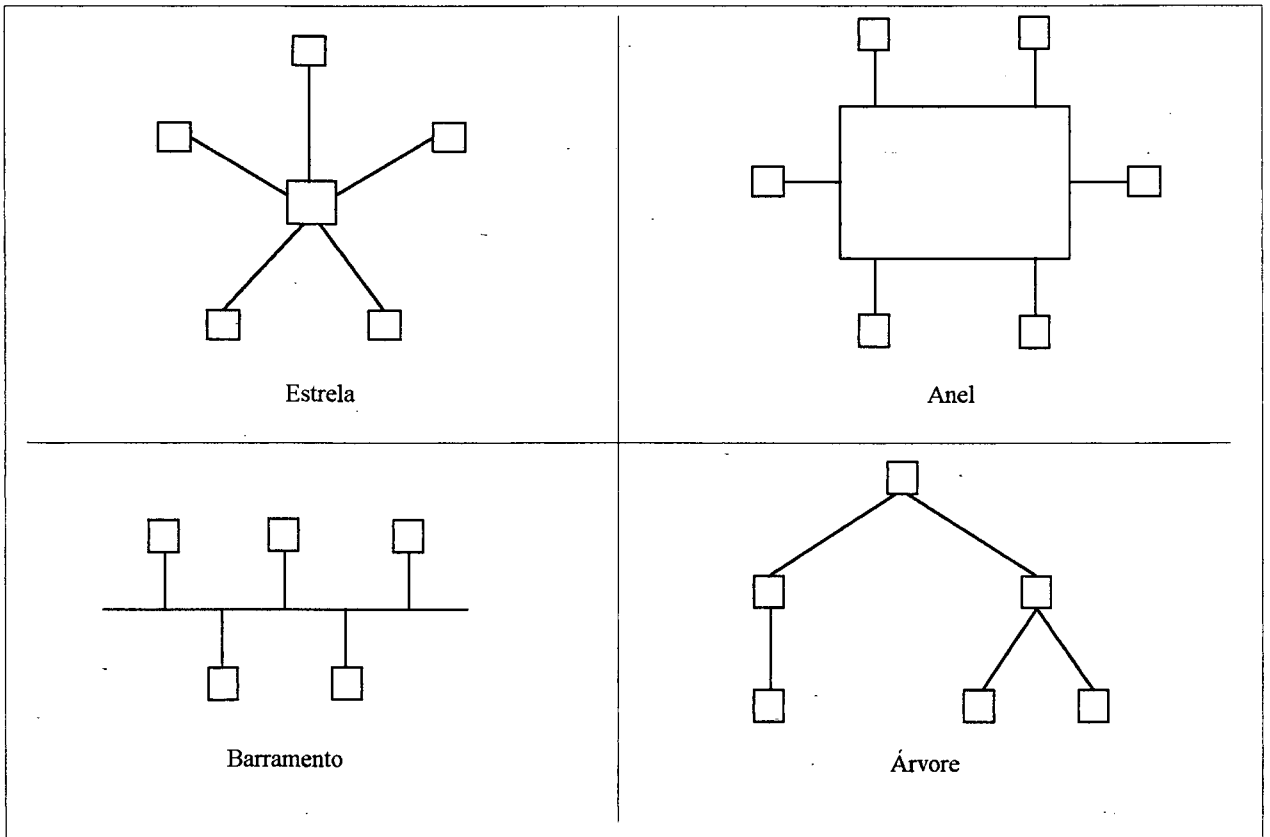


Figura 2.2 - Várias topologias para redes de comunicação

As WANs estão sendo utilizadas para integrar os dispositivos a um nível mais amplo na empresa, denominado nível CIM (figura 2.1). Por outro lado, as LANs estão sendo aplicadas em sistemas FMS e em células flexíveis de manufatura (figuras 2.3 e 2.5, respectivamente).

Através de uma rede de comunicação, os dados da produção (p.ex., *status* de uma determinada máquina, tempos de ciclos, produtividade, falhas ou erros em determinados setores) podem ser rapidamente repassados aos diversos setores da empresa.

Algumas benefícios no uso de LAN nos sistemas flexíveis de manufatura são [Hohner 1989]:

- redução do estoque em processo - WIP;
- ciclos de manufatura mais curtos;
- maior produtividade e qualidade;
- *feedback* da produção para níveis superiores; e
- suporte formal para o controle da produção.

Atualmente, os protocolos que mais estão sendo aplicados em sistemas CIM/FMS/FMC são o MAP (*manufacturing automation protocol)* e o *field bus*. Segundo Aletan (1991), o MAP já é considerado um padrão de rede de "tempo-real" para indústria.

Pelas características, o protocolo MAP é utilizado a nível de CIM e FMS, enquanto o *field bus*, na comunicação a nível de células (FMCs).

2.1.6 - FILOSOFIAS DE MODELAGEM

Várias técnicas de inteligência artificial (AI), redes de Petri (RdP) e redes neurais (NN), estão sendo empregadas no desenvolvimento de sistemas de manufatura (CIM/FMS/FMC). Zhang e Huang (1995), mencionaram que "AI tem provido diversas técnicas

com aplicações na manufatura, com sistemas especialistas baseados no conhecimento sendo um dos mais promissores desenvolvimentos. Recentemente, redes neurais têm sido implementadas com sucesso na manufatura". As redes de Petri também têm sido amplamente utilizadas nesses sistemas, pois o controle de sistemas FMS é complicado e é similar ao controle de sistemas dinâmicos a eventos discretos, os quais podem ser perfeitamente modelados por RdP.

Para demonstrar como AI, RdP e NN estão sendo aplicadas em sistemas de manufatura, descreve-se a seguir algumas referências de trabalhos sendo feitos por profissionais da área:

Inteligência artificial - sistemas especialistas:

- simulação e projeto de sistemas flexíveis de manufatura: [Vujosevic 1994];
- inteligência artificial para engenharia simultânea [Young et al. 1992];
- inteligência artificial na manufatura: [Udo 1992];
- escalonamento (*scheduling*) e funções de controle: [Rabelo e Alptekin 1989];
- aplicação de inteligência artificial no gerenciamento de manutenção: [Ray e Murty 1989];

Redes Neurais[1]:

- projeto de sistemas flexíveis de manufatura: [Chryssolouris et al. 1990];
- projeto de formação de células flexíveis de manufatura através de redes neurais: [Venugopal e Narendran 1994], [Chu 1993], [Rao e Gu 1995], [Lee et al. 1992];

[1] Algumas dessas referências foram obtidas do artigo escrito por Zhang e Huang (1995).

- formação de familias de peças: [Kao e Moon 1991], [Moon e Chi 1992];
- robótica: [Horne et al. 1990];
- monitoração: [Gövekar et al. 1989], [Hou e Lin 1993], [Burke e Rangwala 1991], [Mazory 1991], [Upadhyaya e Eryurek 1992];
- controle de processos na manufatura: [Pugh 1989], [Chryssolouris et al. 1992];
- escalonamento: [Arizono et al. 1992], [Foo e Takefugi 1988], [Chen 1992];
- planejamento do processo: [Osakada e Yang 1991], [Knapp e Wang 1992].

Redes de Petri:

- análise de gerenciamento de ferramentas em FMS: [Reddy et al. 1992];
- avaliação de desempenho de sistemas FMS: [Chan e Wang 1993];
- modelagem e validação de sistemas de manufatura: [Choi et al. 1994];
- uso de modelos abstratos para análise de sistemas flexíveis de manufatura: [Kochikar e Narendran 1994];
- simulação e controle de FMS através de redes de Petri coloridas: [Cossins e Ferreira 1992];
- uma estrutura para medir a flexibilidade de FMS: [Kochikar e Narendran 1992];
- especificação, modelagem e controle de uma célula flexível de manufatura: [Braga 1993], [Huang e Chang 1992];
- modelagem de sistemas de manufatura: [Teng e Zhang 1993].

2.1.7 - PLATAFORMAS PARA IMPLEMENTAÇÃO

Qual a melhor decisão a ser tomada com relação à plataforma sobre a qual um *software* CIM (integrador CIM, gerente FMS ou gerente FMC) executará - DOS, Windows, Windows NT, OS/2 ou UNIX ? A resposta, a princípio, dependerá dos objetivos do sistema. Se o programa estiver voltado para a indústria, numa aplicação real, é aconselhável o uso do sistema UNIX [Hill 1992]; por outro lado, se o trabalho for puramente acadêmico, usar DOS, Windows ou OS/2 é perfeitamento possível. Dentre esses, entretanto, Windows NT e OS/2, da mesma forma que o próprio UNIX, por serem sistemas multitarefas (i.e., capazes de realizar mais de uma tarefa simultaneamente) são mais adequados a esse tipo de aplicação.

Se comparado com UNIX, usar DOS, Windows ou OS/2 pode sair bem mais barato. Mesmo tendo-se que adquirir co-processadores, aceleradores gráficos, placas de comunicação e mais memória, os computadores pessoais (PCs) ainda possuirão um preço bem mais acessível do que uma estação de trabalho. Por outro lado, usar UNIX traz benefícios bastante significativos como, por exemplo, ser multi-usuário (capaz de servir a vários usuários simultaneamente), possuir banco de dados relacionais (SQL), protocolo de comunicação TCP/IP e interface *X Windows*. Entretanto, como os custos envolvidos na implementação CIM são sobremaneira elevados, pode-se, na maioria dos casos, desconsiderar as diferenças de custos entre essas plataformas.

Como visto anteriomente, tanto o OS/2 quanto o Windows NT e o UNIX são sistemas multitarefas. Esta é uma característica muito importante em aplicações CIM. Considere, por exemplo, a situação em que um sistema de controle de uma FMC (gerente FMC) precisa receber do nível hierárquico superior (gerente FMS) uma determinada ordem de processo. Se o gerente FMC for desenvolvido numa plataforma multitarefas, ele não precisará interromper as

atividades de controle dos movimentos do robô para se comunicar com o gerente FMS, pois ele poderá facilmente realizar essas duas tarefas concorrentemente.

2.1.8 - PROGRAMAÇÃO ORIENTADA POR OBJETOS

Deseja-se que um sistema CIM/FMS/FMC seja flexível, para isto ele deve ser capaz de rodar em qualquer classe de computador (*laptops*, PCs e *mainframes*), em diversas plataformas de *hardware* e ser facilmente integrado com outros sistemas que rodam em diferentes níveis herárquicos.

Lopes (1992) disse que a programação orientada por objetos (OOP) permite que o *software* seja *flexível*, pois "é uma saida radical dos sistemas de programação do passado...alterando a relação entre dados e programas usados...encapsulando junto: atributos de dados e procedimentos, gerando um módulo ou objeto". Segundo Perry (1994), a programação orientada por objetos reflete muito mais a vida real do que as linguagens procedurais não-OOP, e suas principais características são sua forma natural de programar e escrever códigos.

A abordagem baseada em programação orientada por objeto está sendo bastante utilizada em sistemas de manufatura, principalmente no desenvolvimento de bancos de dados e sistemas de engenharia. OOP também tem sido aplicado em várias outras áreas, como por exemplo em projeto [Zaremba 1993].

Mayer (1988), citado por Leva et al. (1993), diz que a orientação por objetos traz uma série de vantagens as quais podem significativamente contribuir para o desenvolvimento de sistemas CIM, do ponto de vista da ciência da computação bem como do ponto de vista dos sistemas de engenharia. Estes benefícios incluem: proteção de informações de estruturas internas, capacidade de abstração de dados, herança de propriedades de objetos.

Entretanto, as características mais importantes provindas da utilização da orientação por objetos em sistemas CIM são:

- a relação próxima entre funções e dados de um objeto do sistema (encapsulamento do objeto);
- modularidade do sistema, haja visto que cada objeto é um módulo com suas próprias características e comportamento;
- reusabilidade de objetos existentes (como referência ou modelos parciais), que levam a uma economia de esforços de projeto e de codificação;
- expansibilidade do sistema, pois novos objetos podem ser facilmente adicionados ao sistema existente; e
- a capacidade de evolução do sistema atual para o gerenciamento das mudanças que ocorrem durante o ciclo de vida do sistema (evolução do objeto).

2.2 - SISTEMA FLEXÍVEL DE MANUFATURA

Sistemas flexíveis de manufatura procuram obter uma elevada produtividade junto com uma grande flexibilidade, objetivando satisfazer as demandas dos mercados competitivos de hoje. Eles têm passado por uma rápida evolução e desenvolvimento, sendo instalados numa variedade de configurações, em uma ampla gama de ambientes de manufatura [MacCarthy 1993]. Cada ambiente desses costuma, por vez, fazer suas próprias definições sobre o que é um sistema flexível de manufatura. Hedin (1994), por exemplo, diz que diversas classificações têm sido sugeridas na literatura para sistemas FMS, e exemplifica essa diversidade de conceitos referenciando Browne *et al.* (1994), Rachamadugu and Stecke (1987) e Jaikumar and Van Wassenhove (1989).

Huang e Chang (1992) sugerem que um FMS pode ser decomposto na combinação de diversas FMCs. Cossins e Ferreira (1992), vão um pouco além e dizem que FMS é um grupo de FMCs interconectadas por um sistema de transporte de peças, tudo controlado por um sistema computacional comum. A definição elaborada que será usada neste trabalho é a seguinte:

> *um sistema FMS é formado pela integração de células flexíveis de manufatura, de sistema de transporte de materiais (MHS), de sistema de armazenamento e retirada automático de materiais (AS/RS), de sistemas de engenharia, de gerente de dispositivos de fixação e de ferramentas, através de uma rede de comunicação local (LAN); o seu controle é feito por um software de gerenciamento (gerente FMS), que roda em um computador central e dispõe de um banco de dados (DB).*

Essa definição pode ser melhor entendida através do desenho mostrado na figura 2.3.

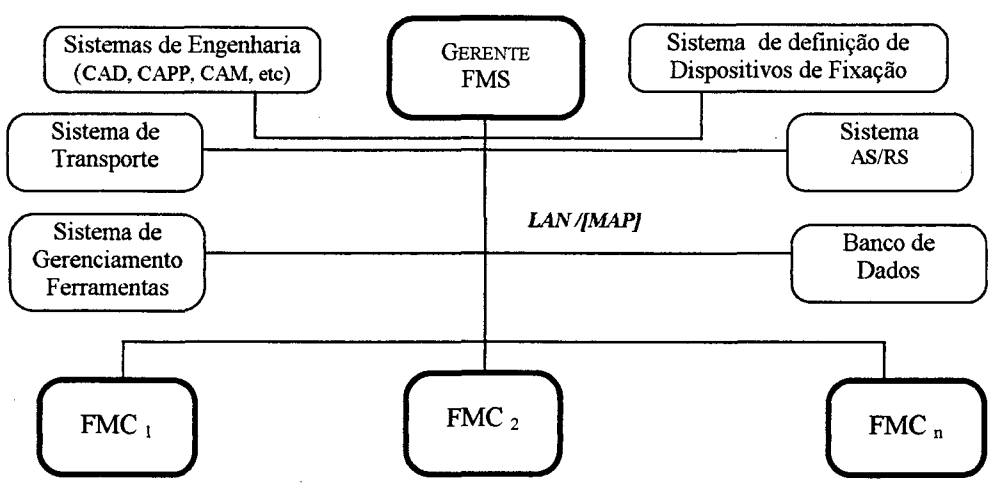


Figura 2.3 - Arquitetura FMS

2.2.1 - O GERENTE FMS

Deseja-se que o sistema FMS seja realmente flexível, i.e., capaz de responder rápido e eficientemente às mudanças de ambiente tais como as variações de demanda, a mudança nas especificações de produto e de qualidade, bem como às situações dinâmicas dentro do próprio sistema, como quebras e bloqueios de máquinas [Kochikar e Narendran 1992]. Por isso são considerados sistemas extremamente complexos. Seu controle precisa ser preciso e rápido o suficiente para corrigir os problemas ocorridos a fim de evitar ineficiências indesejadas [Teng e Zhang 1993].

Para se atingir os objetivos desejados em um sistema flexível de manufatura, é indispensável que o gerente FMS seja muito bem especificado. Devem ser definidas questões como: o sistema operacional a ser utilizado, a linguagem de programação, a rede e o protocolo de comunicação, o grau de confiabilidade e segurança das informações de processo, as técnicas de modelagem, etc. Esses fatores, aliados ao grande número de sistemas heterogêneos que devem ser integrados, fazem do desenvolvimento do *software* de gerenciamento de FMS uma tarefa extremamente difícil.

2.2.1.1 - CRITÉRIOS DE PROJETO DE UM GERENTE FMS

Baseando-se num caso realmente prático, isto é, fora do ambiente acadêmico, Nagarkar e Bennett (1988) sugeriram uma série de critérios para implementação de sistemas de gerenciamento de FMS. Esses critérios, listados a seguir, podem servir não somente como um passo inicial para a elaboração de sistemas FMS, mas também para outros sistemas com aplicação na manufatura:

- o sistema de controle (ou gerente) deve estar voltado para encontrar a melhor maneira de realizar as

especificações do cliente (usuário), com o mínimo de inventário e uso otimizado dos recursos disponíveis;

- o pedido de um cliente só deve ser enviado para a primeira operação, quando todas as peças necessárias para produzir aquele lote estejam disponíveis. Isto garante que o pedido estará completo num tempo mínimo possível;
- o *software* deve ter robustez apropriada para garantir cerca de 100% de integridade e segurança das informações (alta confiabilidade);
- controle da qualidade apropriado na fabricação de produtos e componentes;
- obtenção dos dados e o fluxo de informações sem papel, usando unidades visuais de *display*, leitores de código de barras e *scanners* a laser;
- o sistema deverá garantir a segurança dos operadores. O controle, por exemplo, dos AGVs[1] (veículos guiados automaticamente) no chão de fábrica, deverá prever algum dispositivo de parada de emergência, para o caso, por exemplo, de algum funcionário estar sobre o percurso por onde o AGV irá passar;
- informações da produção continuamente incorporadas ou acessíveis ao programa de gerenciamento. Isto permite, por exemplo, a visualização, em tempo real, do *status* das máquinas, dispositivos e equipamentos.

[1] O AGV é um sistema de transporte de materiais que está sendo muito utilizado em grandes indústrias. Sua operação é razoavelmente complexa e técnicas como redes de Petri têm sido empregadas em seu controle e/ou simulação (Yim e Linn 1993).

2.2.1.2 - ATRIBUIÇÕES DE UM GERENTE FMS

Deseja-se que o gerente do sistema seja capaz de realizar funções como:

- permitir a comunicação com o nível hierárquico superior, para, por exemplo, enviar relatórios de produção e *status* de células, máquinas ou de qualquer outro dispositivo, receber ordens de serviço e planos de produção, etc;
- fazer o planejamento da produção (*scheduling*), que diz respeito principalmente à alocação e ao sequenciamento das células para fabricação de um determinado lote de peças. Este problema é ainda mais complicado quando o número de peças diferentes é muito elevado;
- gerenciar o fluxo de materiais dentro do FMS (peças em *buffers*, estoques intermediários);
- coordenar o tráfego dos dispositivos de transporte de materiais (*MHS*), que pode ser por exemplo um AGV;
- se comunicar com os computadores onde são desenvolvidos os sistemas de engenharia, como: projeto (CAD), planejamento do processo (CAPP), os programas NC (CAM), etc;
- enviar os programas NC para os gerentes das FMCs;
- se comunicar com os sistemas que gerenciam os dispositivos de fixação e as ferramentas;
- monitorar, inicializar e interromper qualquer atividade em qualquer módulo do FMS;
- identificar, através da leitura em um sistema de código de barras ou em um sistema de visão, por exemplo, os tipos de peças a serem produzidas num determinado instante,
- permitir que um novo tipo de peça (ou família de peças) seja incorporado à produção;

- alterar a prioridade dos pedidos, de maneira que se um novo pedido é extremamente urgente, ele deve ser incluído numa posição privilegiada na lista de pedidos;
- emitir alarmes (sonoros e/ou visuais) quando da ocorrência de anormalidades no sistema.

2.2.1.3 - AS INTERFACES DE COMUNICAÇÃO DE UM GERENTE FMS

O gerente FMS deve se comunicar com diferentes níveis hierárquicos dentro de uma fábrica. A um nível hierárquico mais alto o gerente deverá se comunicar com o integrador CIM e/ou um cliente (usuários); no mesmo nível, com os dispositivos já citados anteriormente e demonstrados na figura 2.3, e com o nível inferior, com os gerentes das células. A figura 2.4 mostra estas interfaces, sendo que algumas delas são descritas a seguir.

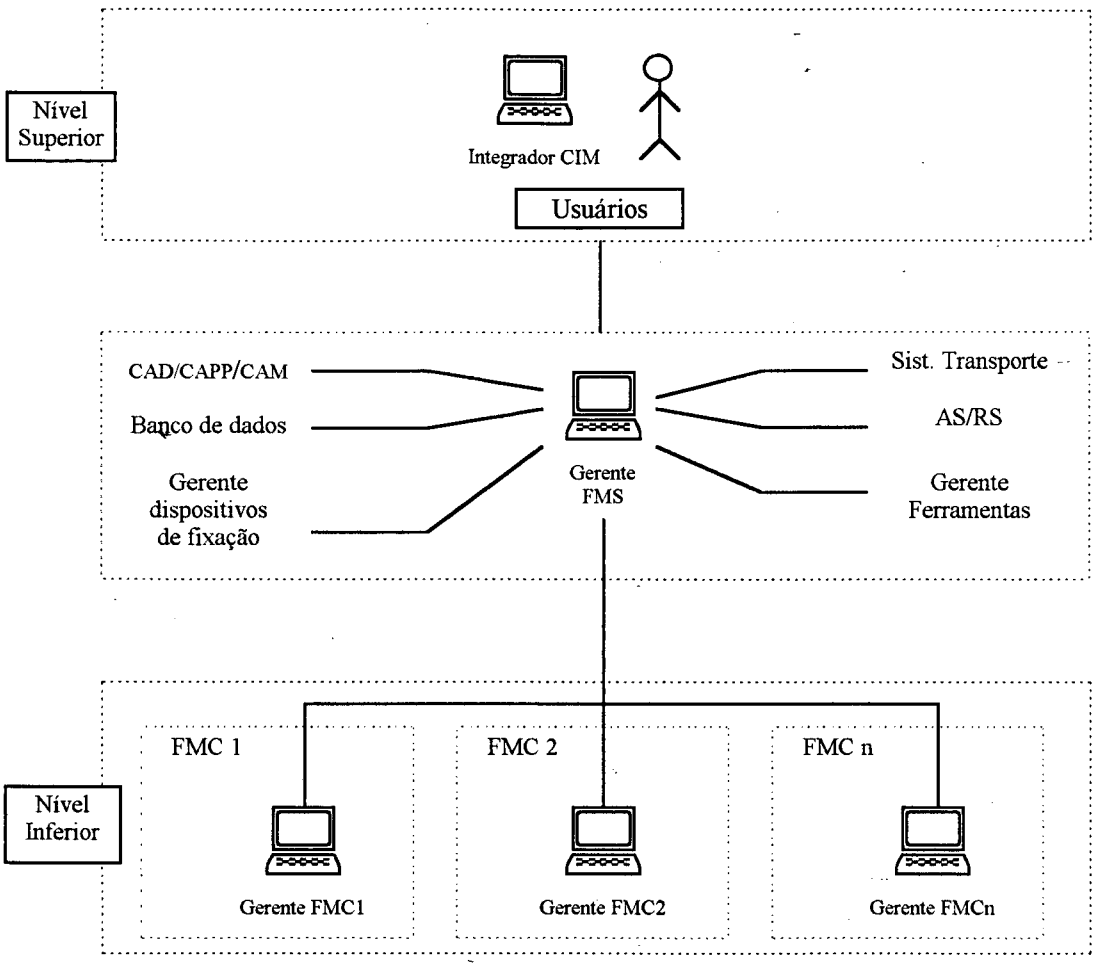


Figura 2.4 - As interfaces de um gerente FMS

Interface gerente FMS↔usuário: O gerente FMS deve atender às especificações do usuário (que pode ser um técnico e/ou integrador CIM).

Interface gerente FMS↔banco de dados: É indispensável que o gerente FMS tenha a seu dispor um banco de dados (DB). Este poderá estar no mesmo computador (onde roda o gerente FMS), num outro, ou então distribuído em vários outros computadores.

É no DB que o gerente FMS armazena todas as informações referentes à produção passada, atual e o que se deseja para o futuro. É nele que ficam armazenadas as informações relativas à

produtividade, *status* da produção, planejamento da produção, listagem dos pedidos, etc, informações estas que podem ser transmitidas para o nível superior (CIM) ou inferior (FMC), dependendo da situação.

Interface gerente FMS↔gerente de dispositivos de fixação: Uma das maiores dificuldades encontradas num sistema flexível é a falta de um *software* que defina automaticamente qual dispositivo de fixação que deve ser usado para cada tipo de peça.

A comunicação entre o gerente FMS e o gerente de dispositivos de fixação (GDF) pode-se dar, por exemplo, através do envio do desenho da peça (p.ex., via arquivo '.DXF') junto com a informação de qual célula deverá produzi-la do gerente FMS para o GDF. O gerente FMS, quando receber do GDF a informação de qual dispositivo de fixação utilizar, deverá fazer com que este seja montado na FMC.

Interface gerente FMS↔sistema de transporte de materiais: Independente de quais sistemas de transporte estão sendo utilizados no FMS (AGVs, *conveyors*, esteiras,etc), o gerente deverá poder acompanhar os seus movimentos pelo sistema. O controle de colisões entre AGVs e/ou entre AGVs e pessoas, por exemplo, é importante e deve ser muito bem implementado (no sistema de transporte).

Interface gerente FMS↔sistema AS/RS: O controle de quantas peças (brutas, acabadas, intermediárias) estão armazenadas no sistema é de extrema importância dentro dos sistemas flexíveis. Hoje, na era do *just-in-time* e da forte concorrência, quem tem menos estoque, menos perde.

Também foi colocado como um critério de projeto (seção 2.2.1.1) que o pedido de um cliente só deve ser enviado para a primeira operação, quando todas as peças necessárias para produzir aquele pedido estejam disponíveis. Portanto, o sistema de armazenamento e retirada automático de materiais (AS/RS) deve estar integrado com o gerente FMS.

Interface gerente FMS↔sistemas de engenharia: Uma indústria que quer se manter competitiva no mercado mundial tem que utilizar o computador no projeto (CAD), no planejamento do processo (CAPP), na manufatura (CAM), etc, e por isso é fundamental que o gerente FMS esteja integrado com esses "sistemas de engenharia" (ESs).

Como os sistemas de engenharia precisam de informações sobre máquinas, dispositivos e ferramentas disponíveis na fábrica, o gerente deve também permitir a troca de informações entre os ESs e os gerentes de dispositivos de fixação e de ferramentas. É essa intregração que possibilita o desenvolvimento da engenharia concorrente.

Interface gerente FMS↔gerente de ferramentas: Um sistema FMS torna-se ainda mais complexo devido à existência de uma grande quantidade e variedade de ferramentas. Para isto, existem sistemas dedicados exclusivamente ao gerenciamento desses dispositivos, chamados de gerentes de ferramentas. Um eficiente sistema de gerenciamento de ferramentas pode trazer grandes benefícios à indústria, tais como [Reddy et al. 1992]:

- garantia do funcionamento ininterrupto do sistema;
- maior lucratividade;
- redução dos tempos de produção em 30%; e
- redução dos tempos de preparação das ferramentas em 70%.

2.3 - CÉLULA FLEXÍVEL DE MANUFATURA

Uma célula flexível de manufatura (FMC) é o módulo do FMS onde há efetivamente manufatura de produtos, é a unidade básica de produção. Ela é o elemento chave na implementação de métodos da manufatura flexível [Huang e Chang 1992].

Uma FMC é formada a partir da integração de máquinas, dispositivos de manipulação e movimentação de peças (p.ex., robôs, esteiras), sistemas de medição (para inspeção/controle da qualidade), magazines ou *buffers* (onde são armazenados peças ou estoques de peças), e um computador central, onde executa o supervisor ou *software* de gerenciamento da célula (gerente FMC). Um exemplo de FMC está ilustrado na figura 2.5.

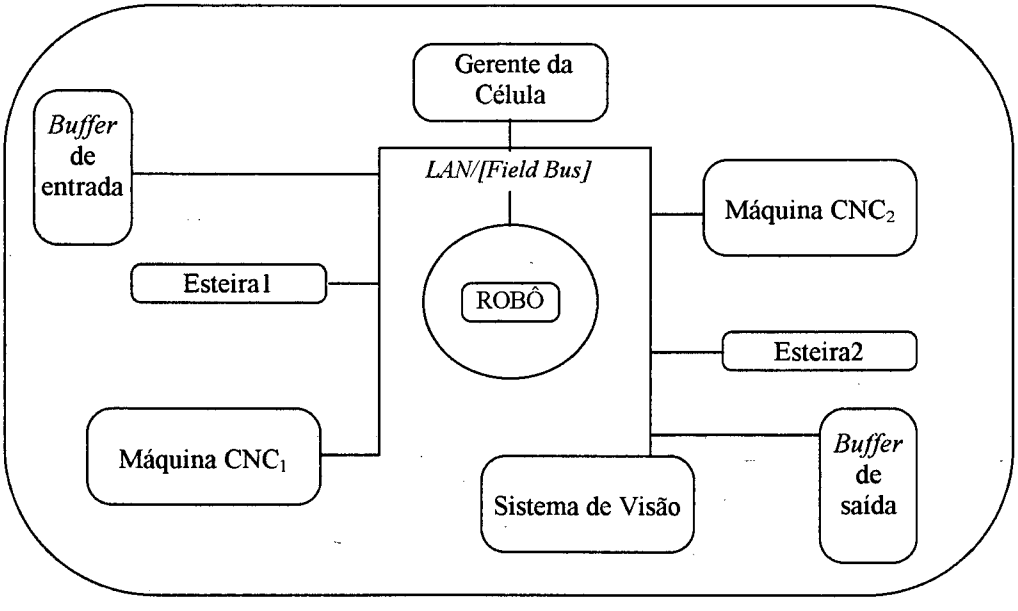


Figura 2.5 - Um exemplo de FMC

A integração dos dispositivos numa célula é feita por meio de uma rede de comunicação local, utilizando um outro tipo de protocolo de comunicação, geralmente um *field bus*.

Algumas aplicações bastante comuns das células flexíveis de manufatura são em operações como: usinagem, pintura, montagem e soldagem.

Devido à sua flexibilidade, as FMCs têm favorecido a implantação de estratégias inovadoras de gerenciamento da

produção nesses últimos tempos, como o *just-in-time* (JIT), o controle da qualidade total (TQC) e a Tecnologia de Grupo (TG).

Segundo Welke e Overbeeke (1988), a manufatura celular é um dos melhores veículos para se implementar o *just-in-time* e o controle da qualidade total. Para Knight e Wall (1989), o uso da tecnologia de grupo melhora a comunicação entre os grupos de trabalhadores em células flexíveis de manufatura. Segundo eles, os operadores de uma célula com TG podem mais eficientemente gerenciar suas áreas, planejar suas atividades, controlar seus processos, reconhecer os problemas e encontrar soluções.

Algumas das vantagens das células flexíveis de manufatura são [Welke e Overbeek 1988]:

- racionalização do transporte de material;
- menos refugo, melhor qualidade;
- ambiente de trabalho mais estimulante;
- redução/eliminação de áreas de armazenamento de estoques intermediários;
- melhoramento no planejamento da produção;
- maior flexibilidade;
- redução no tamanho dos lotes;
- redução no tempo de produção.

Algumas dessas vantagens são obtidas diretamente devido ao uso das estratégias de gerenciamento mencionadas anteriormente (JIT, TQC, TG), as quais também são repassadas para os níveis hierárquicos superiores, tornando-se vantagens CIM/FMS.

Pode-se dizer que a relação entre FMC, FMS e CIM se dá como ilustrada na figura 2.6, isto é, uma FMC é considerada um sub-conjunto (módulo) de um FMS, que por sua vez é um sub-conjunto da manufatura integrada por computador.

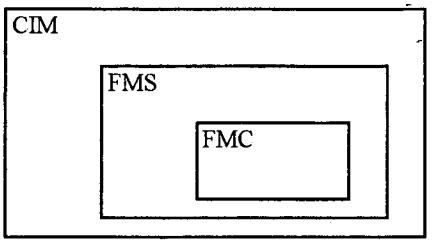


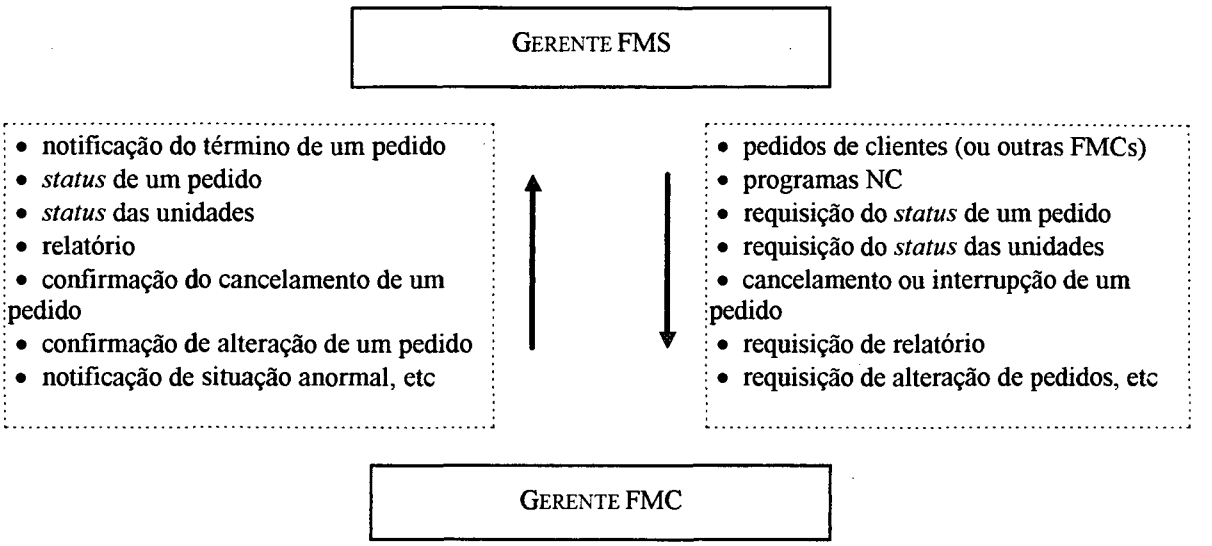
Figura 2.6 - CIM-FMS-FMC

2.3.1 - O GERENTE FMC

É no computador central da célula que roda o *software* de gerenciamento da FMC, ou simplesmente, o gerente FMC.

De acordo com Culbreth e Pollpeter (1988), este *software*, também chamado de "supervisor" ou "controlador" da célula, é responsável por todo o planejamento da produção, alocação de recursos (máquinas, ferramental e mão-de-obra) e coordenação de tráfego dentro da célula. É ele quem envia os programas para as máquinas e para o(s) robô(s), faz o controle de falhas e erros na produção, recebe do nível hierárquico superior (gerente FMS) as ordens de processo, pedido de *status* dos equipamentos, relatórios de produtividade, enfim, responsabilidades relativas à FMC são suas atribuições.

Baseando-se em [Braga 1993], um fluxo de informações entre um gerente FMC e um gerente FMS poderia ser por exemplo:



Algumas atribuições típicas dos gerentes FMC que se enquadram dentro das definições deste trabalho são [Rehg 1994]:

- monitoração da produção;
- monitoração do processo;
- monitoração dos equipamentos;
- distribuição dos programas;
- gerenciamento de avisos e alarmes;
- controle estatístico da qualidade e do processo;
- entrada de dados e de eventos;
- distribuição de tarefas e escalonamento;
- controle e monitoração da trajetória da ferramenta;
- controle e acompanhamento dos estoques na FMC;
- geração de relatórios sobre as atividades da célula;
- diagnóstico de um problema na célula;
- auxílio ao operador; e
- programação *off-line* do sistema.

2.4 - RESUMO

Muitos e variados conceitos de CIM, FMS e FMC estão sendo aplicados em indústrias e instituições de ensino. Este capitulo procurou mostrar um pouco sobre essas diversidades de modelos, mas, principalmente, tentou clarear um pouco esse emaranhado de idéias, estabelecendo divisões e conceituações próprias.

Foram vistas definições bem diferentes das desenvolvidas aqui, como por exemplo, autores que consideram um FMS como sendo uma evolução de uma FMC, outros ainda que incluem os sistemas de engenharia (CAD,CAPP,CAM,etc) como sendo módulos do CIM. O importante é concluir que ainda não se tem uma definição padronizada para CIM, FMS e FMC.

Foram descritos também algumas características e atribuições dos *softwares* que regem as atividades nos sistemas CIM/FMS/FMC.

Modernas técnicas computacionais como inteligência artificial, redes neurais, rede de Petri e programação orientada por objetos, junto com algumas topologias e protocolos de comunicação que estão sendo aplicadas em sistemas de manufatura, foram também mencionadas neste capítulo.

A partir do próximo capítulo, será descrito o desenvolvimento do trabalho prático, que como será observado, está centrado no desenvolvimento de um sistema produtivo que simula um sistema de manufatura composto por duas células flexíveis de manufatura.

3

DESCRIÇÃO GERAL DO TRABALHO

Este capítulo procura situar o leitor dentro do contexto do trabalho proposto, fornecendo uma visão mais elaborada sobre seus objetivos e características desejadas, além de descrever sobre algumas ferramentas que serão utilizadas tanto na implementação da FMC1 como da FMC2.

3.1 - OBJETIVOS E CARACTERÍSTICAS

O objetivo do trabalho proposto é implementar e integrar duas células flexíveis de manufatura, junto com o desenvolvimento dos seus respectivos *softwares* de gerenciamento, criando-se assim um pequeno sistema produtivo (SP) possível de ser encontrado em indústrias automatizadas.

O SP proposto, além das células acima descritas, deverá ser composto também por uma terceira FMC. Entretanto, como ela já está em funcionamento como resultado de um trabalho anterior, a descrição de sua implementação não fará parte dos objetivos do presente trabalho.

A primeira célula (FMC1) será responsável pela fabricação de peças (ou componentes), por isso pode ser chamada de "célula de usinagem". A segunda célula (FMC2) terá a função de montar essas peças para formar o produto final, e pode ser chamada de "célula de montagem".

A FMC1 também será responsável pelo transporte das peças usinadas à célula de montagem, que por sua vez, deverá transportar o produto montado à terceira célula. Na FMC3, deverão ser executadas operações como pintura, soldagem, etc[1].

Neste sistema, o produto, quando sair da FMC3, poderá ser diretamente colocado no mercado.

Pode-se mencionar também que em nenhum momento da produção haverá intervenção humana, o que significa dizer que todas as operações para a produção do produto poderão ser executadas "com as luzes apagadas" (ou num turno à noite).

Conforme o conceito adotado no capítulo anterior, este SP proposto não poderá ser considerado um sistema flexível de manufatura, principalmente porque suas células não estarão integradas ao nível hierárquico superior (gerente FMS) e não existirão, por exemplo, módulos como os sistemas de engenharia (CAD,CAPP,CAM,etc), o sistema de armazenamento e retirada automático de materiais (AS/RS), considerados essenciais a qualquer sistema do tipo FMS.

Todos os equipamentos a serem utilizados no sistema foram construídos exclusivamente para fins acadêmicos, como os robôs e as esteiras, por exemplo. Como infelizmente a UFSC não dispõe de máquinas CNC didáticas, o trabalho proposto ficará limitado e um tanto quanto simplificado, pois as mesmas terão que ser simuladas.

Por isso, o objetivo do trabalho proposto é implementar um sistema que simula um SP que pode ser encontrado em indústrias automatizadas.

Como a usinagem não existirá de fato, não haverá necessidade de se ter nas FMCs um sistema de controle (ou um banco de dados) das ferramentas a serem utilizados na fabricação das peças.

---

[1] Deve-se lembrar que as operações (pintura, soldagem, etc) são também simuladas.

Também não se poderá ter uma máquina real para inspeção e controle da qualidade das peças fabricadas e por isso, essa máquina também deverá ser simulada. O mesmo já ocorre com as máquinas da FMC3, cujas operações se darão através de esperas (paradas) em *estações de trabalho*.

Quanto à estratégia de planejamento e controle da produção, o sistema produtivo poderá ser classificado tanto como um sistema que **empurra a produção para o mercado** (tipo *"push"*), quanto um sistema onde a **produção é puxada pelo mercado** (tipo *"pull"*) [ver BRAGA, 1993, pág. 8]. Quem de fato deverá decidir sobre esta questão é o gerente FMS.

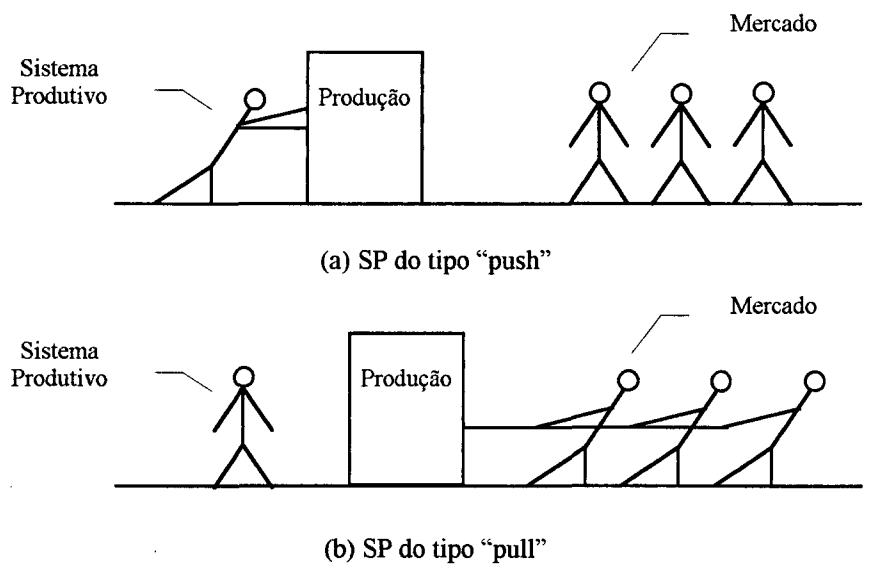


Figura 3.1 - Estratégias de planejamento e controle da produção

Um outro ponto importante e que faz parte da delimitação da área de abrangência do trabalho proposto diz respeito à utilização do conceito de produção com *estoque zero*, independente se o controle do fluxo da produção se dará por KANBAN (para sistema do tipo *"pull"*) ou MRP (para sistemas do tipo *"push"*).

Assim a célula de usinagem não deverá ter nenhum estoque intermediário e, quando uma peça for considerada ruim, o gerente deverá dar prioridade à produção deste tipo de peça novamente, até que uma peça boa possa ser enviada à célula de montagem. Esta é uma consideração um tanto quanto simplificativa e desaconselhável na prática, porque faz com que a célula de montagem fique parada, esperando a chegada de uma peça boa.

3.2 - FERRAMENTAS COMUNS ÀS CÉLULAS

Nesta seção e também no próximo capitulo, serão tratados três assuntos referentes a algumas ferramentas a serem utilizadas no desenvolvimento dos *softwares* de gerenciamento das células. São elas:

- O controlador MARK-IV e o seu *driver* de comunicação;
- redes de Petri interpretadas; e
- o pacote gráfico - INWIN.

3.2.1 - O CONTROLADOR E O *DRIVER* DE COMUNICAÇÃO

Conhecer o controlador MARK-IV e poder utilizá-lo perfeitamente (através de seu *driver* de comunicação) é de grande importância para que se possa atingir os objetivos do trabalho proposto. Entretanto, estes assuntos não serão tratados neste ponto da dissertação, pois, como exigem um certo grau de detalhamento, há a necessidade de se ter um capítulo exclusivamente para este propósito.

Com o intuito de situar o leitor neste contexto, pode-se dizer que o MARK-IV é um controlador desenvolvido pela *Rhino Robotics Corporation* e tem como função principal controlar o

robô XR4 ou SCARA, também desenvolvidos por aquela empresa. Já o *driver* de comunicação é um módulo que teve que ser desenvolvido para permitir que um determinado aplicativo (ou programa hospedeiro) possa comandar, via comunicação serial, o controlador MARK-IV.

3.2.2 - REDES DE PETRI

Al-Jaar e Desrochers (1989) e Peterson (1981), citados em [Choi et al. 1994], dizem que redes de Petri (RdP) têm sido usadas com sucesso para modelar, controlar e analisar sistemas dinâmicos a eventos discretos, caracterizados por paralelismo, processos assíncronos, "*deadlocks*" (bloqueios "mortais"), conflitos e processos dirigidos por eventos ("*event-driven processes*"). Eles afirmam também que redes de Petri fornecem modelos precisos e métodos eficientes de análise, pois: (1) capturam interações tanto de eventos paralelos quanto sequenciais, (2) podem ser derivadas do conhecimento de como o sistema opera, (3) fornecem modelos concisos para situações de conflitos e de controle de tamanhos de *buffers* e (4) permitem a implementação de análises em tempo-real.

As RdP também estão sendo muito usadas para controle de sistemas de manufatura que, de acordo com Teng e Zhang (1993), são considerados complicados e similares aos sistemas dinâmicos a eventos discretos. Além disso, eles dizem que o controle de uma célula flexível de manufatura, por exemplo, situa-se dentro do contexto de controle de sistemas de manufatura.

Uma rede de Petri é uma ferramenta simples para modelagem do fluxo de peças num sistema de manufatura e para representação de execuções ordenadas de operações individuais [Rembold et al. 1993].

Optou-se então por utilizar esta ferramenta no gerenciamento das FMCs a serem desenvolvidas no trabalho proposto.

As redes a serem utilizadas nos gerentes das células serão do tipo interpretadas (RPI), isto porque o autor já possui um certo conhecimento nesses tipos de redes e inclusive já desenvolveu um *software* para esses tipos de RdPs (*rede*)[1].

Deverão ser utilizadas apenas redes de Petri interpretadas, entretanto outros tipos de RdPs poderiam ser utilizadas, como por exemplo, redes de Petri coloridas, que estão sendo empregadas em sistemas de manufatura, para simular e controlar FMSs [Cossins e Ferreira, 1992], e especificar, modelar e controlar FMCs [Huang e Chang, 1992].

Como o objetivo do trabalho não está centrado na utilização de redes de Petri em sistemas FMC, este assunto não será abordado em profundidade. Para se obter informações mais completas sobre os fundamentos de redes de Petri, o leitor poderá recorrer a: [Murata, 1989], [Chan e Wang, 1993] e [Cossins e Ferreira, 1992]. Em [Choi et al. 1994] é mostrado um exemplo de modelagem de uma FMC por redes de Petri coloridas.

3.2.2.1 - UMA RPI NO CONTEXTO DO TRABALHO

Através de um exemplo, mostrado na figura 3.2, será explicado como uma RPI será utilizada no gerenciamento das células e como ela deverá ser escrita para poder ser lida pelo gerente da célula.

---

[1] O *software* de RPI (*rede*) a ser utilizado nos gerentes das células, não faz parte do desenvolvimento do trabalho proposto. Ele foi desenvolvido pelo autor em 1993, quando estava trabalhando como bolsista de iniciação científica no Laboratório de Controle e Microinformática Marcos Cardoso Filho (LCMI/UFSC), sob orientação do então mestrando Francisco de Assis Soares e da Professora Janete Cardoso.

Nesta rede-exemplo, se houver uma peça no *buffer* de entrada [*Peça_Buffer_Entrada*] e o robô e a máquina estiverem disponiveis [*Robô_Disponível* e *Máquina_Disponível*], o robô deverá pegar a peça e levá-la à máquina [*Robô_Levando_Peça*]. Quando a peça estiver na posição correta dentro da máquina, o robô deverá abrir a garra, deixando a peça na máquina [*Peça_Na_Máquina*]. Em seguida, deverá retornar à sua posição inicial [*Robô_Retornando*] liberando a peça para ser usinada [*Peça_Pronta_Para_Usinagem*]. Enquanto ele retorna, a máquina já inicia a usinagem da peça [*Máquina_Usinando*].

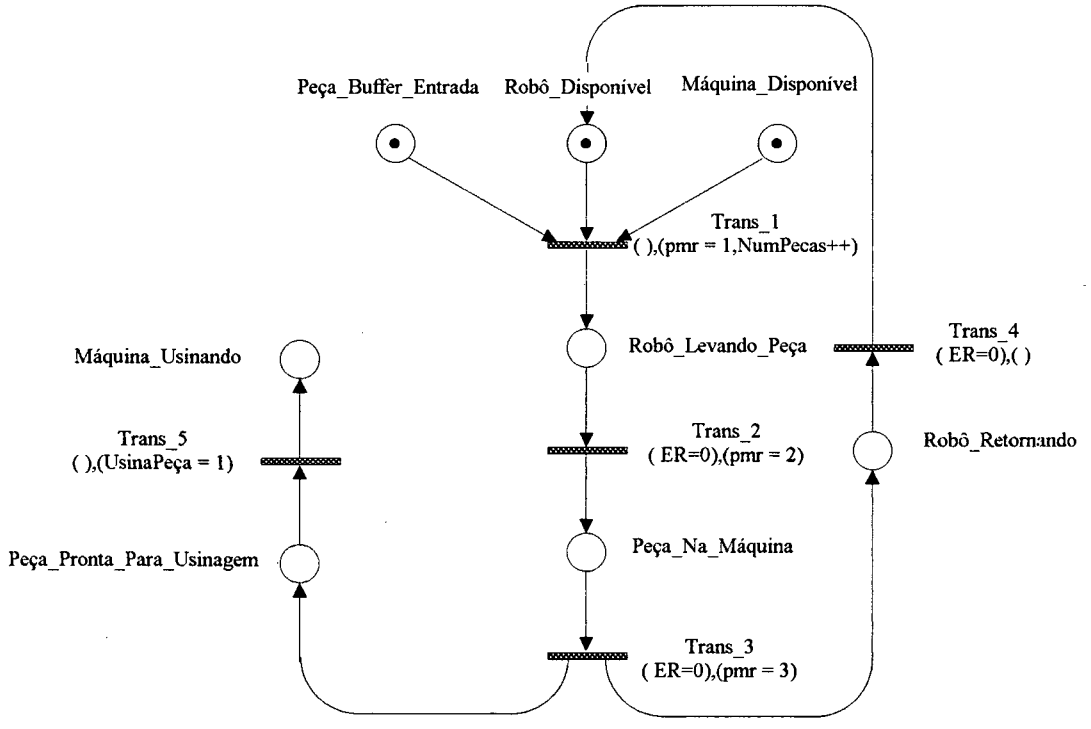


Figura 3.2 - Uma RPI exemplo

Esta rede, mesmo servindo como um exemplo, fornece uma boa idéia do que deverá ser encontrado nas RPIs dos gerentes das células.

Esta rede é composta por:

- 8 lugares: Peça_Buffer_Entrada,

Robô_Disponível,
Máquina_Disponível,
Robô_Levando_Peça,
Peça_Na_Máquina,
Peça_Pronta_Para_Usinagem,
Robô_Retornando, e
Máquina_Usinando;

- 5 transições: Trans_1,
Trans_2,
Trans_3,
Trans_4, e
Trans_5;
- 4 variáveis: pmr (próximo movimento do robô),
ER (Estado do robô, 0:parado e 1:movendo-se),
NumPeças (número de peças), e
UsinaPeça.

Como se pode observar, todas as transições da rede possuem variáveis associadas, representadas no desenho por dois grupos de parênteses abaixo no nome da transição. Conforme estabelecido no *software* da rede (*rede*), essas variáveis, que são sempre do tipo inteiro, podem ser internas ou externas. As variáveis do tipo internas só podem ser acessadas pela rede, entretanto as do tipo externas podem ser acessadas por outros programas fora do software rede. Será visto no capítulo 5 que essas varíaveis externas têm um papel muito importante no gerenciamento das células, pois são elas que farão o interfaceamento entre os dispositivos das célula (sensores, esteiras,etc) e o gerente.

No primeiro grupo de parênteses, encontram-se as condições. Se todas as condições não estiverem satisfeitas, a transição, mesmo estando sensibilizada, não poderá disparar.

Nestes casos, diz-se que a transição está sensibilizada mas não está habilitada.

Assim que uma transição habilitada é disparada, uma série de ações associadas a ela é executada. Essas ações são formadas pelo conjunto de variáveis que compõem o segundo par de parênteses.

Uma variável pode atuar como condição e ação numa mesma transição simultaneamente.

As condições podem ser do tipo:

- "=" (igual a),
- "!=" (diferente de),
- "<" (menor que),
- "<=" (menor ou igual a),
- ">" (maior que), e
- ">=" (maior ou igual a).

As ações podem ser:

- "=" (atribuição de um novo valor),
- "+=" (soma de um valor),
- "-=" (subtração de um valor), e
- "*=" (multiplicação por um valor).

No caso da transição TRANS_2 por exemplo, se ela estiver sensibilizada, só poderá disparar quando o robô estiver parado (ER=0). Quando TRANS_2 disparar, o robô deverá realizar o movimento número 2 (pmr=2), que neste caso significa que ele deve abrir a garra.

Para se poder entrar com uma nova RPI no gerente da célula (no caso de se querer produzir um novo tipo de peça, por exemplo), deve-se escrevê-la sob uma sintaxe específica, em forma de um arquivo texto. Para se carregar a rede do exemplo acima, por exemplo, deve-se escrevê-la da seguinte maneira:

```
REDE     nome_da_rede;
NODOS
         Peça_Buffer_Entrada, Robô_Disponível,
         Máquina_Disponível                    : LUGAR(1);{número de fichas = 1}
         Robô_Levando_Peça, Peça_Na_Máquina,
         Peça_Pronta_Para_Usinagem, Robô_Retornando,
         Máquina_Usinando                      : LUGAR;{número de fichas = 0}
         Trans_1, Trans_2, Trans_3, Trans_4, Trans_5      : TRANSIÇÃO;
ESTRUTURA
         Trans_1:(Peça_Buffer_Entrada,Robô_Disponivel,Máquina_Disponivel),
         (Robô_Levando_Peça);
         Trans_2:(Robô_Levando_Peça),(Peça_Na_Máquina);
         Trans_3:(Peça_Na_Máquina),(Peça_Pronta_Para_Usinagem,Robô_Retornando);
         Trans_4:(Robô_Retornando),(Robô_Disponível);
         Trans_5:(Peça_Pronta_Para_Usinagem),(Máquina_Usinando);
FIM
VARIÁVEIS  {considerando-se as variáveis como sendo externas e de valor inicial
         =0}
         pmr, ER, NumPecas, UsinaPeca                : EXTERNA(0);
ETIQUETAS
         Trans_1:( ),(pmr=1, NumPeças++);
         Trans_2:(ER=0),(pmr=2);
         Trans_3:(ER=0),(pmr=3);
         Trans_4:(ER=0),( );
         Trans_5:( ),(UsinaPeça=1);
FIM_REDE;
```

Também está incluso no *software "rede"* o **jogador de fichas.** É ele quem dispara uma transição habilitada, removendo as fichas dos lugares de entrada, alocando-as nos lugares de saída e executando as ações associadas. Existem três maneiras de se disparar uma transição habilitada: aleatoriamente (qualquer transição pode disparar), por prioridade (transições mais prioritárias disparam primeiro) ou por ordem de escrita (como num sistema *first-in first-out* (FIFO), as primeiras declaradas disparam primeiro). Nos gerentes das FMCs, esta será a maneira utilizada.

3.2.3 - A INTERFACE GRÁFICA INWIN

Para que os gerentes das células tivessem uma interface com o usuário o mais amigável possível, com grande disponibilidade de recursos, pensou-se em utilizar o ambiente *Microsoft* WINDOWS[1] no seu desenvolvimento. Entretanto, a não disponibilidade de placas multi-seriais nos microcomputadores utilizados fez com que esta idéia fosse descartada pois, o gerente da célula de montagem teria que utilizar as duas portas seriais disponíveis no PC. Desta maneira, o programa não mais admitiria o uso do *mouse*, tornando o WINDOWS praticamente sem vantagem, se comparado com uma interface DOS.

Uma segunda possibilidade era o uso de um pacote gráfico desenvolvido pela *Borland International* chamado TURBO VISION (TV). Este pacote possibilita a utilização de uma interface gráfica muito semelhante às interfaces utilizadas em linguagens de programação com ambientes turbo, como o Turbo PASCAL, Turbo C, etc).

Entretanto, como será explicado nos capítulos seguintes, o gerente de uma célula tem que atender tanto as funções de atendimento à célula (FAC) quanto ao usuário (FAU), e deve portanto ser planejado de forma a executar um *"pseudo" paralelismo* entre as FAC e as FAU. Infelizmente o TV não permite que esse paralelismo aconteça pois ele exige "cem por cento" da atenção, não permitindo a execução alternada entre duas sub-rotinas.

---

[1] O ambiente Microsoft WINDOWS, apesar de ainda não ter sido publicamente declarado, já é quase um padrão em termos de desenvolvimento de *software* para PC. No caso de sistemas de manufatura, *softwares* de CAD/CAM, por exemplo, estão sendo lançados com suas últimas versões aptas para rodar somente em WINDOWS.

Talvez uma saida para este problema fosse a utilização de interrupção, mas não se seguiu por este caminho pois, de qualquer forma o uso do *mouse* estaria impossibilitado, como no caso do WINDOWS.

Considerando-se os pontos acima e a questão que o gerente FMC deve atender principalmente a célula e não ao usuário, optou-se por desenvolver uma interface própria, chamada de INWIN.

3.2.3.1 - CARACTERÍSTICAS DE IMPLEMENTAÇÃO DO INWIN

O INWIN foi implentado utilizando-se a filosofia de programação orientada a objetos (OOP). Ele não permite a utilização do *mouse* e trabalha apenas com a parte gráfica em modo texto. Dentre suas principais características, tem-se:

- permite o desenvolvimento de janelas de apresentação (JA) e menus (M);
- possibilita utilizar moldura (*frame*) simples ou dupla;
- permite até 23 linhas (em modo JA) ou 23 opções (em modo M);
- permite ao usuário selecionar as cores de fundo e frente da moldura, das linhas/opções e da opção selecionada; e
- um menu pode ser do tipo vertical ou horizontal.

Como o INWIN foi implementado através de OOP, o usuário poderá verificar que facilmente pode criar, mover ou alterar uma opção (ou linha) num menu ou numa janela. No anexo I é feita uma declaração das funções do INWIN utilizadas nos gerentes FMC.

3.3 - RESUMO

Este capitulo procurou mostrar os objetivos a serem atingidos pelo trabalho proposto junto com uma sucinta descrição sobre algumas ferramentas que serão utilizadas no gerenciamento das FMCs, como: redes de Petri interpretada, seu *software* (*rede*) e a interface gráfica desenvolvida (INWIN). Outras duas ferramentas que também serão utilizadas nas FMC1 e FMC2 são o controlador MARK-IV e o seu *driver* de comunicação, os quais serão tratados no capítulo seguinte, uma vez que eles merecem uma atenção mais detalhada.

4

O CONTROLADOR MARK-IV

Este capitulo tem como objetivo principal descrever o controlador MARK-IV, os robôs XR4 e SCARA, e os *drivers* de comunicação, uma vez que são elementos importantes no desenvolvimento do trabalho proposto.

É com o controlador MARK-IV que o gerente da célula tem que se comunicar. Através dessa comunicação, o *software* poderá controlar o robô (XR4 ou SCARA), as portas de entrada e saída, as esteiras transportadoras (portas auxiliares), as estações de montagem e o *"teach pendant"*. A figura 4.1 mostra onde são feitas as conexões no controlador MARK-IV. Como será visto mais adiante, a integração dos componentes da célula não é feita de fato por meio do seu computador central, mas sim através do MARK-IV.

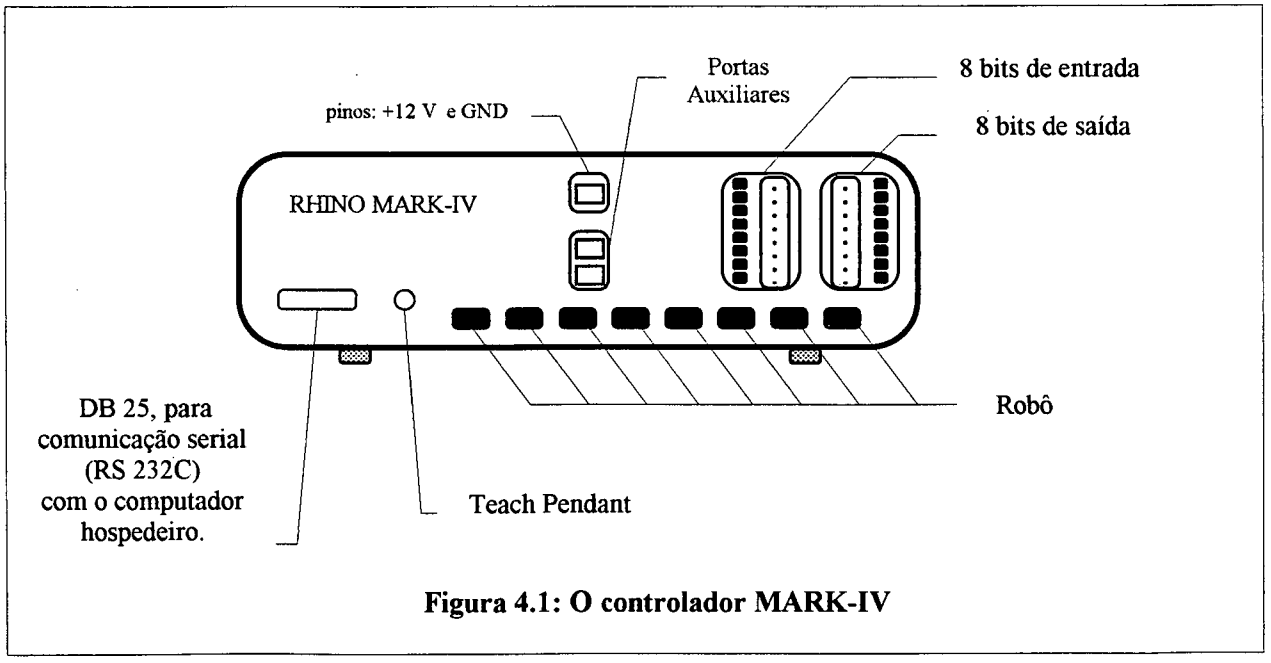


Figura 4.1: O controlador MARK-IV

4.1 - OS DISPOSITIVOS CONECTADOS AO MARK-IV

Antes de se descrever como foi desenvolvido os algorítmos que fazem a comunicação entre o *software* de gerenciamento de uma FMC e o MARK-IV, é importante mostrar as principais características dos dispositivos que são conectados a ele.

4.1.1 - OS ROBÔS XR4 E SCARA

O controlador MARK-IV pode controlar um robô do tipo XR4, SCARA ou genérico (GENERIC). Neste trabalho foram utilizados apenas robôs XR4 e SCARA, e por isso nada será mencionado sobre o tipo GENERIC.

Em [RHINO Robots 1989], o XR4 é descrito como sendo um robô com cinco eixos de coordenadas de revolução com garra elétrica. Pode ser extendido até 57.15 cm e tem capacidade de levantamento de até 998.7 gramas.

O SCARA por sua vez é descrito como um robô com quatro eixos, com garra elétrica, com alcance máximo de 45.72 cm e com capacidade de levantamento de até 454 gramas.

As figuras 4.2 e 4.3 mostram as dimensões e as "juntas" onde atuam os motores nos robôs XR4 e SCARA respectivamente. Algumas fotos nos capítulos 5 e 6 retratam melhor sobre esses robôs.

Segundo [RHINO Robots 1989], todos os eixos de ambos os robôs são controlados por servo motores DC usando *encoders*[1] ópticos incrementais para realimentação ("*feedback*"), para controle de posição, velocidade e aceleração do motor. As garras também possuem a combinação servo motor DC/*encoder*.

---

[1] o *encoder* fornece um sinal de realimentação para o MARK-IV que permite o controle (de posição, velocidade e aceleração) do motor.

Os principais eixos possuem chaves limites (ou chaves fim-de-curso) que são usadas para posicionar o robô numa posição de zero absoluto conhecida (*hard home*[1]). Ambos os braços foram construídos em alumínio para fornecer rigidez com pouco peso.

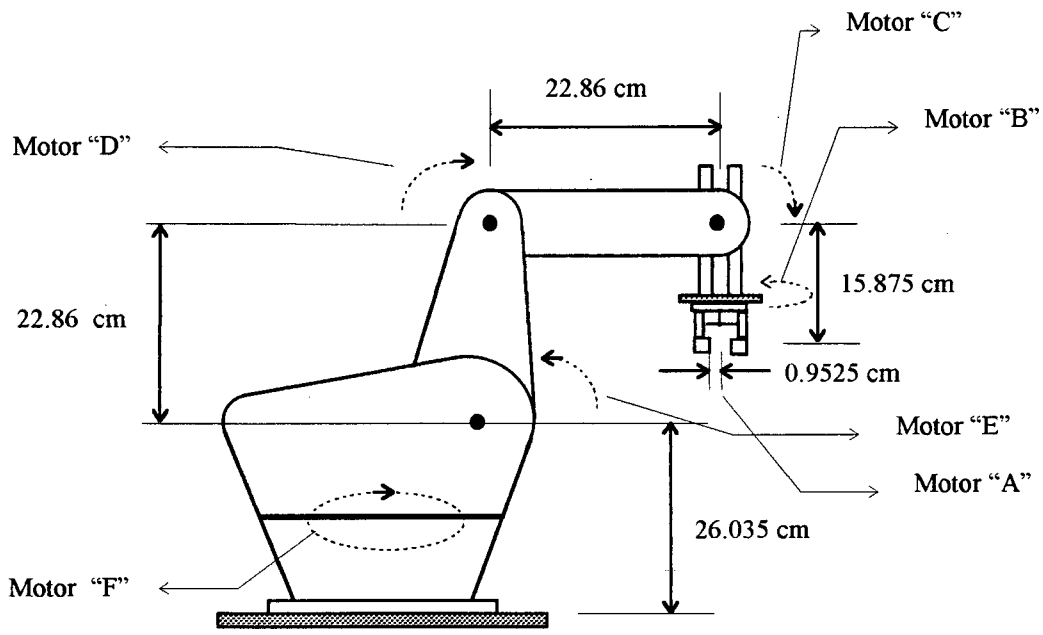


Figura 4.2 - O robô XR4

[1] O *hard home* (HH) de um robô é uma posição constante, fixa, definida pela própria construção do robô. Teoricamente, esse ponto é sempre o mesmo, e por esta razão o usuário pode utilizá-lo como um ponto de referência.

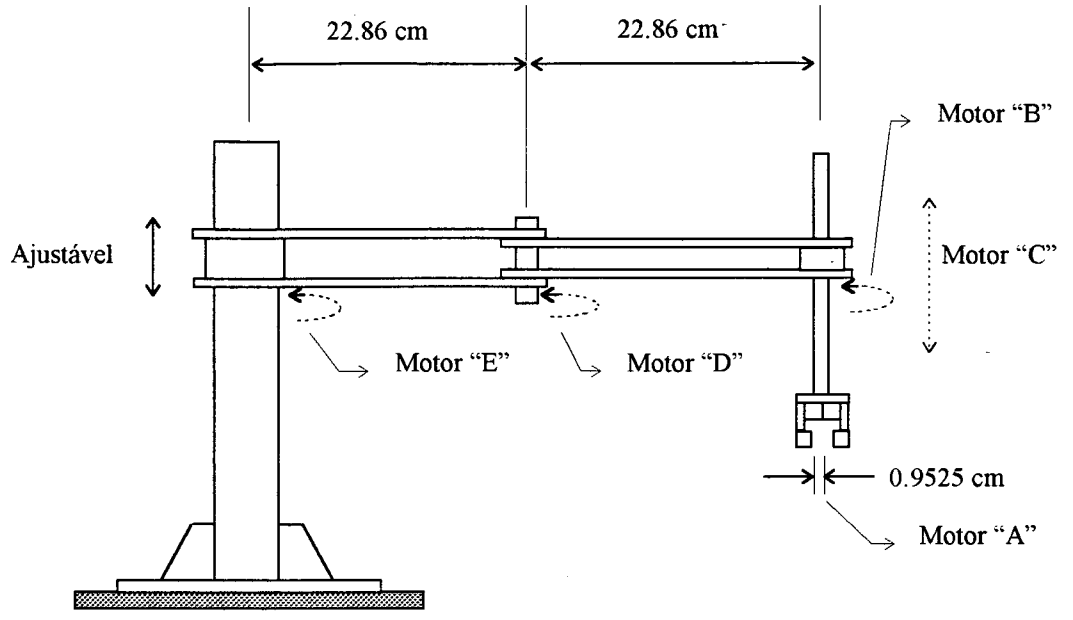


Figura 4.3 - O robô SCARA

Para se obter informações mais detalhadas sobre os robôs XR4 e SCARA, deve-se recorrer a [Rhino Robots, 1991] e [Rhino Robots, 1989].

4.1.2 - O *TEACH PENDANT*

Existem duas maneiras de se trabalhar com o robô: através de um programa hospedeiro (que nesta aplicação é o próprio gerente da célula) e/ou por meio do *teach pendant*.

O *teach pendant* (TP) permite que o usuário altere diretamente a configuração do MARK-IV e, consequentemente, do robô, das portas de entrada e saída, das portas auxiliares, etc. O TP também possui um "ambiente" próprio de programação, que possibilita ao usúario desenvolver e armazenar programas.

Dentre outras vantagens do TP, cita-se:

Biblioteca Universitária
UFSC

- útil para auxiliar a manutenção e o reparo de programas (p.ex., durante o desenvolvimento do gerente, muitas vezes o TP foi utilizado para atualizar os pontos ensinados);
- possui o botão de parada de emergência que, independentemente de se estar ou não trabalhando por meio de um programa hospedeiro, sempre pode ser acionado.

Outras vantagens do TP e melhores informações sobre ele podem ser obtidas de [Rhino Robots, 1989].

4.1.3 - AS PORTAS DE ENTRADA E SAÍDA

O MARK-IV possui um conjunto de oito entradas e oito saídas, denominadas portas de entrada e portas de saída respectivamente, que podem ser conectadas com uma ampla variedade de dispositivos, como por exemplo, sensores industriais.

Esses sinais de entrada e saída são sinais de comando (baixa corrente) opticamente limitados em 40 V e 100 mA.

Se durante o desenvolvimento do trabalho a quantidade de dispositivos a serem integrados na célula ultrapassasse a quantidade de entradas e saídas disponíveis, poder-se-ia fazer uma expansão dessa capacidade por meio de multiplexação (para aumentar o número de entradas) ou através de "*latches*" (para aumentar o número de saídas), conforme detalhado em [Rhino Robots, 1989, páginas 10-4 à 10-6].

4.1.4 - AS PORTAS AUXILIARES: AUX1 E AUX2

As portas auxiliares (AUX1 e AUX2) possibilitam a conecção de até dois motores DC sem encoders, sob corrente de até 3 A e tensão de 12 V.

Através delas pode-se aplicar voltagem e polaridade programável, mas, como não permitem motores com realimentação,

são úteis apenas para aplicações onde o controle (de posição, velocidade, etc) não é importante.

Neste trabalho, essas portas foram utilizadas para alimentar as esteiras transportadoras, as portas de entrada e as portas de saída.

4.2 - A COMUNICAÇÃO DO GERENTE FMC COM O MARK-IV

Em ambas as células implementadas (FMC1 e FMC2), não houve a necessidade de se ter uma rede de comunicação local (LAN) para fazer a integração do gerente com os periféricos da célula (máquinas, esteiras, sensores). Isto porque, como foi dito, o controlador possui um conjunto de entradas, saídas e portas auxiliares que permitem que esses dispositivos externos sejam a ele conectados.

Quando, por exemplo, o gerente quer saber sobre o estado de um determinado sensor na célula, este interroga o MARK-IV, que por sua vez verifica em suas conexões o estado daquele sensor, retornando então a resposta ao gerente.

4.2.1 - O *HARDWARE* E PARÂMETROS PARA COMUNICAÇÃO

Como é estabelecida realmente a comunicação entre o gerente e o controlador? Primeiramente é necessário que haja um *hardware* específico que, acompanhado de certos parâmetros, permite que o programa hospedeiro (PH) se comunique com o controlador.

Esta ligação entre o microcomputador (PC) e o controlador MARK-IV é feita por meio de um cabo de comunicação serial assíncrono RS232-C, sendo a ligação dos pinos conforme ilustrado a seguir.

Host PC (onde roda o Gerente)			**MARK-IV**
pinos (DB25 [fêmea]):			pinos (DB-25 [macho]):
TxD	2	———	2 RxD
RxD	3	———	3 TxD
RTS	4	———	4 CTS
CTS	5	———	5 RTS
DSR	6	———	6 +12V
Signal GND	7	———	7 Signal GND
CD	8	———	8 +12V

Esta configuração serve para conectar o controlador com um PC via porta serial DB25. Entretanto, caso esta porta seja do tipo DB9 (com 9 pinos), a configuração deve ser:

Host PC			**MARK-IV**
pinos (DB9 [fêmea]):			pinos (DB-25 [macho]):
TxD	3	———	2 RxD
RxD	2	———	3 TxD
RTS	7	———	4 CTS
CTS	8	———	5 RTS
DSR	6	———	6 +12V
Signal GND	5	———	7 Signal GND
CD	1	———	8 +12V

Os parâmetros de comunicação exigidos, devem ser:

- velocidade : **9600 bps**
- bits de dados : **7**
- bits de parada : **2**
- paridade : **par**

Depois de estabelecida a ligação fisica entre o controlador e o computador, pode-se partir para o

desenvolvimento dos algoritmos que estabelecerão a comunicação entre eles.

4.2.2 - O PROTOCOLO PARA COMUNICAÇÃO

O protocolo desenvolvido pelo fabricante, utilizado para estabelecer a comunicação entre um programa hospedeiro e o MARK-IV, funciona da seguinte maneira: o PH envia um comando para o MARK-IV; se este comando não precisar de resposta, nada acontece, mas se uma resposta deve ser retornada ao PH, então este deve entrar num *loop* para receber os sinais que vierem do controlador (ver figura 4.7).

A maneira como é feita a comunicação com o controlador se assemelha de certa forma à linguagem de programação *assembly*, pois os comandos são representados também por mnemônicos, como em *assembly* (veja anexo II). Muitos desses mnemônicos devem vir acompanhados do nome do motor (A, B, C, etc) e de um número, como por exemplo:

(a) quando o PH precisar saber qual a posição do motor "F", deve ser enviado ao controlador o seguinte comando: "PA,F"; e

(b) para colocar a próxima posição do motor "F" em -32456 (absoluto), envia-se: "PD,F,-32456". Neste caso, a unidade de posição utilizada é *unidade de contagem de encoder*.

Quanto à função que executam, os comandos se dividem em 7 grupos[1]:

- comandos de estado do sistema;
- comandos de configuração;
- comandos para leitura de motor;
- comandos para atuar no motor;
- comandos de controle do *teach pendant*;

- comandos de ganho; e
- comandos de entrada e saida.

Quanto à resposta de cada comando, tem-se uma outra sub-divisão:

- comando sem resposta;
- comando com resposta do tipo número inteiro;
- comando com resposta do tipo número real;
- comando com resposta do tipo *"string"* (ou frase); e
- comando com resposta do tipo *"byte"*, onde, na realidade, a resposta está nos oito bits que o compõem.

Agora, independente de qual comando está sendo enviado/recebido ao/pelo controlador, os mnemônicos circulam sempre sob a forma de uma sequência de "caracteres" (*chars*).

Tanto nos comandos quanto nas respostas, a sequência de *chars* sempre é finalizada com pelo menos um dos dois "caracteres" de **fim de comando**: '\r' (*carriage return*) e o '\n' (*line feed*). Assim, se o gerente de uma FMC precisar que o robô pare (no caso de uma parada de emergência, por exemplo), ele deve enviar: "MA\r\n" e não apenas: "MA", pois neste caso o comando não será recebido. Fazendo-se analogia às linguagens de programação existentes, pode-se dizer que esta é uma comunicação de "baixo-nível".

De fato no sistema desenvolvido o gerente não se comunica com o controlador neste nível. Foram desenvolvidos *drivers* de comunicação, que realizam essa interface entre o programa hospedeiro (gerente FMC) e o controlador.

---

[1] Para maiores detalhes sobre esses comandos, leia capítulo 4 de [Rhino Robots, 1989].

O gerente, na realidade, não envia os *chars* 'M', 'A', '\r' e '\n' para o controlador. Ele simplesmente "diz" para o *driver* do robô: "pare tudo!", e o *driver* se encarrega de enviar corretamente as informações para o controlador.

4.2.3 - Os *DRIVERS XR4* e *SCARA*

Seria mais coerente que os *drivers* de comunicação já acompanhassem os respectivos controladores quando da aquisição destes. Todavia, isto não ocorre com o presente controlador, que por não possuir este programa, obriga cada aplicativo a desenvolver seu próprio programa de comunicação.

Neste trabalho entretanto, foi implementado um *driver* que poderá ser utilizado por qualquer aplicativo feito em linguagem C++, o que facilitará bastante o desenvolvimento de trabalhos futuros.

A figura 4.4 mostra a relação entre o gerente da célula (PH) e o controlador MARK-IV.

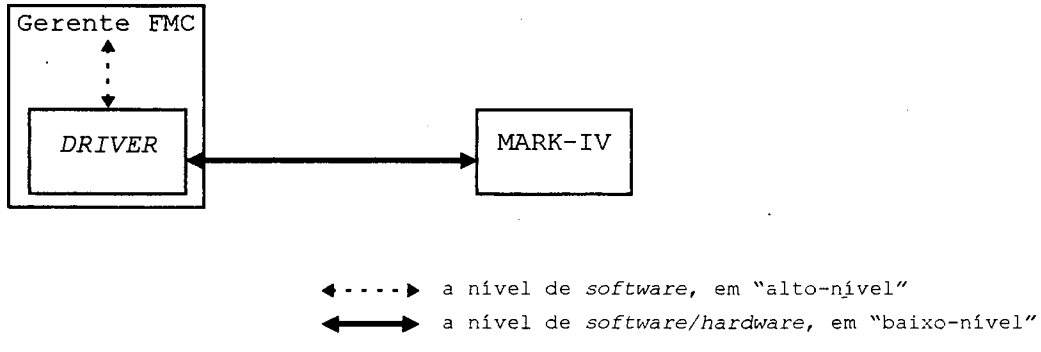


Figura 4.4 - Interface: Gerente FMC↔Mark-IV

Tanto para o caso do robô XR4 quanto para o SCARA, os *drivers* de comunicação são muito parecidos. A principal diferença entre eles é que o *driver* do XR4 (chamado também de *xr4*) precisa levar em consideração que o robô possui um motor a

mais do que o robô SCARA (motor "F"). Fora esta consideração, eles são exatamente iguais, e portanto, serão aqui tratados de uma única maneira. Uma outra generalização que está sendo feita, mas que ainda não foi claramente observada, acontece quando um determinado assunto está se referindo ao "gerente da célula". Neste caso, refere-se tanto ao gerente da célula 1 como da célula 2, indistintamente.

A linguagem de programação escolhida para o desenvolvimento deste trabalho foi "C++", por isso os *drivers* também foram desenvolvidos com esta linguagem. Além disto, eles foram implementados através de programação orientada a objetos, que como fora visto, também foi utilizada para implementar as janelas e menus da interface gráfica dos gerentes e as máquinas da célula de fabricação (capítulo 5). Como se sabe, OOP melhora a modularidade, a estruturação, facilita a compreensão, aprendizagem e a expansão de programas.

A estrutura interna do *driver* é ilustrada na figura 4.5:

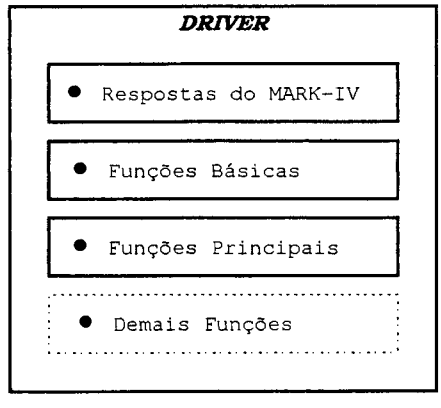


Figura 4.5 - Estrutura básica do *driver* do controlador MARK-IV

O campo das **respostas do MARK-IV** é o lugar onde, se o usuário desejar, poderá acessar as variáveis que guardam os

diferentes tipos de respostas vindas do controlador. Desta forma, a variável *"respInt"*, quando acessada, fornecerá a última resposta do tipo número inteiro enviada pelo controlador; *"respFloat"*, a última resposta real; *"resp"*: a última resposta do tipo *string*; e *"bits[8]"*: a última resposta do tipo *byte*.

O campo das **funções básicas** são as funções que realmente são utilizadas para comunicar o *driver* com o MARK-IV. Nelas são implementados os comandos citados na seção 4.2.2, que são:

- ENVCOM - envia comando e não recebe resposta alguma do controlador;
- ENVCOMRECINT - envia comando e recebe resposta do tipo número inteiro;
- ENVCOMRECFLOAT - envia comando e recebe resposta do tipo número real;
- ENVCOMRECTEXTO - envia comando e recebe *string*;
- ENVCOMRECBYTE - envia comando e recebe *byte*.

O campo das **funções principais** são aquelas que traduzem as funções básicas (de "baixo-nível"), em funções interpretáveis ao programa hospedeiro (de "alto-nível").

As **demais funções** que existem, mas que não foram implementadas, podem ser acessadas através das funções básicas e dos campos de respostas.

Como exemplo do funcionamento do *driver*, supõe-se que o gerente queira saber qual a posição do motor "C" do robô SCARA. Para isto ele faz: **scara.readActualPosition("C", &posC)**. Assim, se nenhum erro ocorreu durante a comunicação, a variável 'posC', anexada pelo seu endereço de memória [&], conterá o valor da posição do motor "C" (ver figura 4.6).

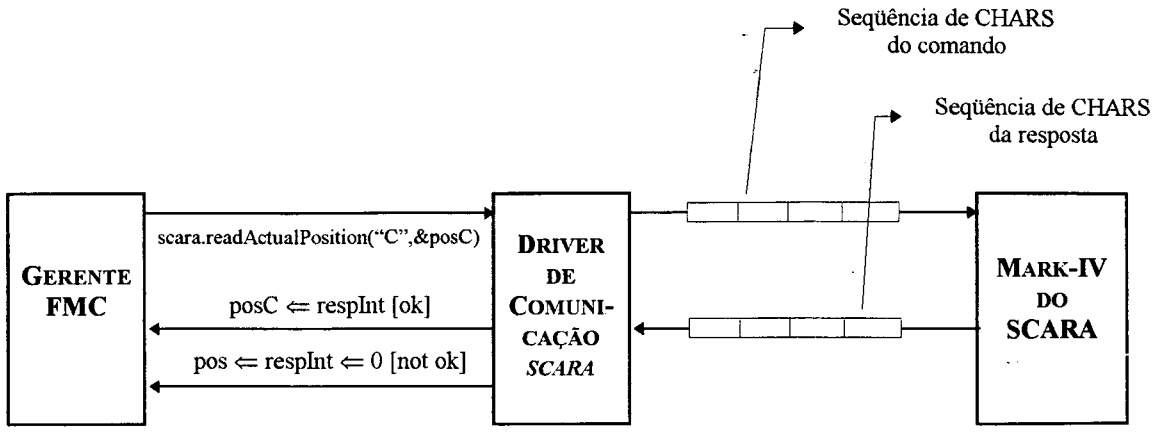


Figura 4.6 - Um exemplo de comunicação entre gerente e o MARK-IV

4.2.3.1 - A FUNÇÃO DE "BAIXO-NÍVEL"

A estrutura geral das funções de "baixo-nível" desenvolvidas no decorrer deste trabalho pode ser melhor entendida através do diagrama mostrado na figura 4.7.

A variável "CONTADOR" teve que ser incluída no algorítmo para fazer o controle de erro durante a comunicação. Inicialmente, CONTADOR era incrementado pela própria função de comunicação, entretanto, mesmo sendo quase desprezível, o tempo gasto para incrementá-lo fazia com que o próximo CHAR que chegava ao *driver* não fosse recebido, pois era perdido, encoberto pelo próximo *char* que vinha logo após ele (reveja a sequência de CHARS mostradas na figura 4.6). Desta forma a informação recebida pelo *driver* estava sempre incorreta. Se o CONTADOR não fosse incluído, quando ocorresse um erro, o *driver* ficaria num *loop* infinito e a única solução seria desligar o computador.

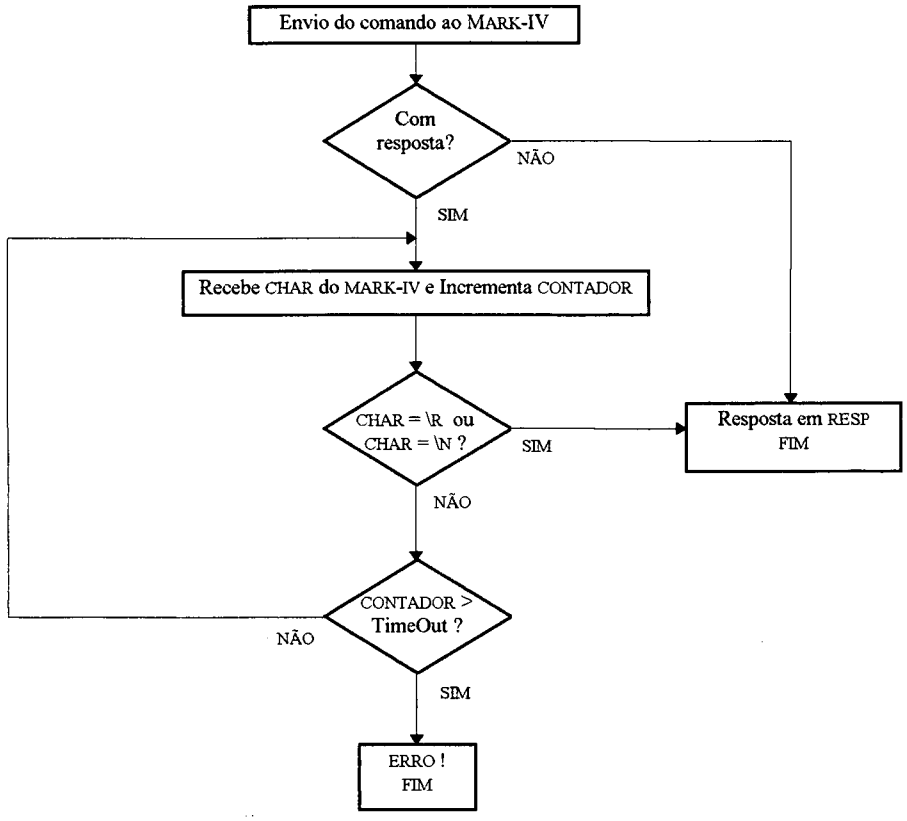


Figura 4.7 - Função de comunicação do *driver* com o MARK-IV

A melhor solução encontrada foi implementar o incremento do CONTADOR através de uma outra função acessada por *interrupção*. Assim, a cada 18 milisegundos CONTADOR é incrementado e nenhuma informação é perdida.

Se o valor de CONTADOR atingir um valor máximo (*TimeOut*), então ocorreu um problema ou mesmo uma perda de comunicação, e facilmente pode-se sair do *loop* da função de comunicação.

Como as informações disponíveis nos manuais e catálogos do MARK-IV não são suficientes, muitos outros problemas ocorreram. Após vários testes, descobriu-se que para

a perfeita comunicação do *driver* com o MARK-IV duas considerações tinham que ser estabelecidas a priori:

(1°) é necessário se ter alguns *delays*[1] distribuídos em diversos pontos das funções de comunicação;

(2°) um comando deve ser enviado ao controlador pelo menos duas vezes, para aumentar a confiabilidade da comunicação, "obrigando-o" a receber o comando. Esta não é uma condição indispensável, como a primeira consideração, mas em comados como parada de emergência, por exemplo, ela é muito aconselhável.

4.3 - RESUMO

Este capítulo procurou fornecer ao leitor um conhecimento a respeito do *hardware* do controlador dos robôs e dos *drivers* de comunicação desenvolvidos. São os *drivers* que permitem que seja feita a comunicação entre os gerentes das células e os seus respectivos dispositivos.

Durtante a fase de implementação do *driver*, vários problemas só foram resolvidos após uma série de tentativas experimentais, pois pouca, ou quase nenhuma, referência está disponível àqueles que desejam desenvolver tais sistemas.

Acredita-se que uma das contribuições do trabalho proposto é o fato de que aqueles que desejarem desenvolver aplicativos com o controlador MARK-IV e os demais dispositivos (robôs, esteiras, etc), poderão utilizar-se dos *drivers* de comunicação desenvolvidos neste trabalho e/ou das informações descritas neste capítulo.

---

1 *Delay*: pequeno intervalo de tempo em que o processador do PC fica parado

5

A Célula Flexível de Fabricação

Após o desenvolvimento dos *drivers* de comunicação (*xr4* e *scara*), deu-se início à implementação da célula flexível de fabricação (FMC1).

No sistema produtivo proposto, esta célula é responsável pela produção das peças e, da maneira como foi definida, o único processo de fabricação a ser utilizado será a usinagem.

Deverão ser produzidas peças rotacionais (cilíndricas) e/ou prismáticas, e por isso a célula será composta de duas máquinas ferramenta: um torno CNC (para fabricar as peças rotacionais) e um centro de usinagem CNC (para produzir as peças prismáticas).

Após a fabricação, as peças deverão ser inspecionadas por um sistema de visão (SV). Através da inspeção, uma peça poderá ser considerada do tipo BOA ou RUIM. Por simplificação, não foram consideradas peças do tipo RETRABALHO, que seria um caso intermediário, onde a peça não está pronta para ir para a próxima célula, nem está tão ruim que deva ser jogada fora (refugo).

Se a peça for considerada BOA, o robô a transportará para a célula de montagem, caso contrário, ele a levará para um *buffer* de refugo.

Como as máquinas CNC e o sistema de visão não existem de fato (elas são simuladas conforme visto no capítulo 3), elas serão chamadas por outros nomes, para tornar o trabalho um pouco mais realista. Sendo assim, a máquina ferramenta torno CNC será chamada de *dispositivo_Torno*, o centro de usinagem CNC, *dispositivo_Centro* e a máquina de inspeção, *dispositivo_SV*.

O fato de não existirem máquinas na FMC1 faz com que as peças a serem fabricadas tenham que entrar no sistema já prontas. Porém, mesmo não havendo usinagem de fato, sabe-se que os objetivos do trabalho poderão ser atingidos.

O sistema produtivo proposto simulará um SP capaz de produzir quaisquer tipos de produtos, desde que os mesmos possam ser fabricados por meio de um torno CNC e/ou centro de usinagem CNC. Neste trabalho entretanto, será produzido apenas um tipo de produto (figura 5.1), formado a partir da montagem de três peças rotacionais (figuras 5.2 à 5.4) e uma prismática (figura 5.5).

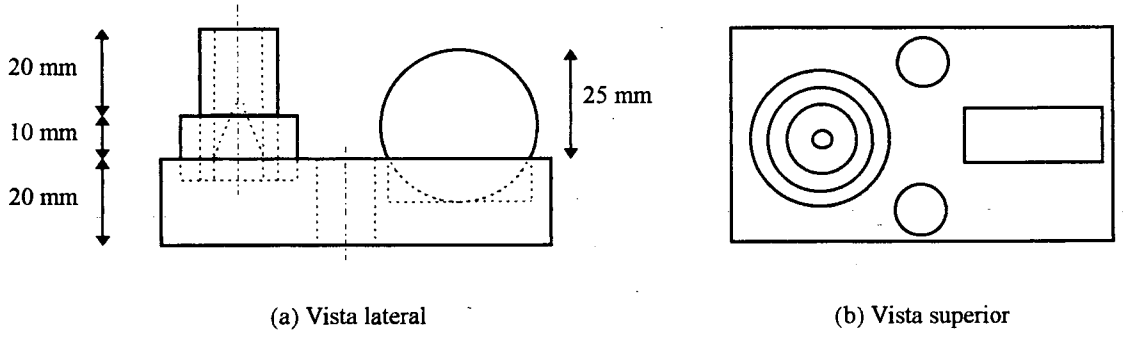


Figura 5.1 - Produto final

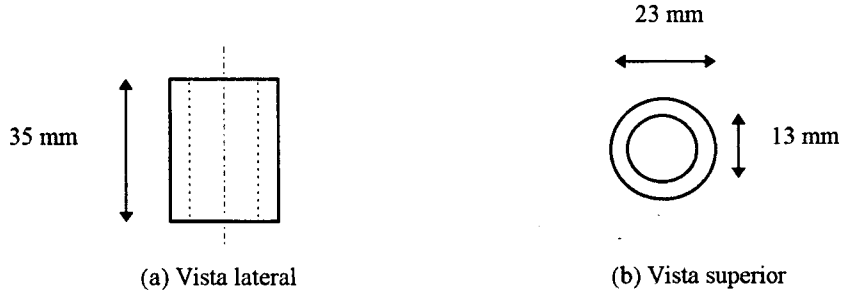


Figura 5.2 - Peça rotacional do tipo 1

Figura 5.3 - Peça rotacional do tipo 2

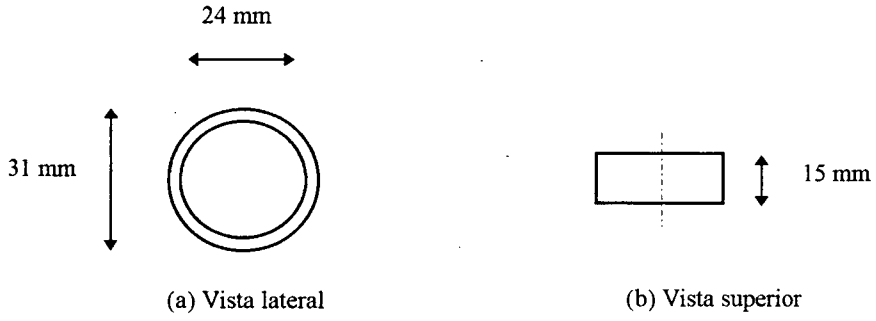


Figura 5.4 - Peça rotacional do tipo 3

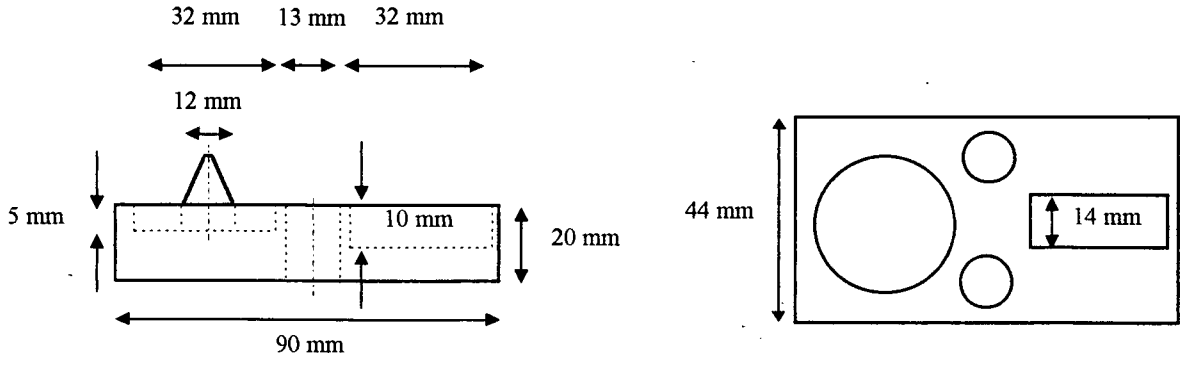


Figura 5.5 - Peça prismática

Se a FMC1 fosse uma célula industrial, isto é, com máquinas reais, haveria também a necessidade de se ter uma rede de comunicação local para integrar os componentes ao computador central da célula. Nesta FMC entretanto, o controlador do robô é quem faz o papel da LAN.

5.1 - O DISPOSITIVO_MÁQUINA

A simulação do torno, do centro de usinagem e do sistema de visão é feita por um mecanismo chamado "dispositivo_Máquina". Esse dispositivo é formado por dois elementos: um *hardware* e um *software*.

A parte de *hardware* representa fisicamente a máquina, servindo principalmente como um local onde o robô pode depositar a peça.

Na FMC1, todos os "dispositivos_Máquina" foram feitos de madeira com uma pequena base plana, que é o local onde deve permanecer a peça a ser trabalhada. Cada dispositivo também possue um *led* (pequena lâmpada), que mostra ao usuário (ou operador) da célula o estado da máquina naquele momento.

A parte de *software* do dispositivo_Máquina é implementada por um programa. Esse programa (ou módulo), faz a simulação do comportamento da máquina.

O comportamento de uma máquina real é bastante complexo, entretanto sua modelagem é feita pelo dispositivo_Máquina de maneira bastante simplificada. Nesta simulação, uma máquina pode se encontrar em quatro estados possíveis:

(1) DISPONÍVEL (LIVRE),
(2) TRABALHANDO (PROCESSANDO),
(3) COM ATIVIDADE ACABADA, e
(4) COM PROBLEMAS(OU QUEBRADA).

O usuário poderá saber qual o estado atual de uma máquina da célula observando o estado do seu *led*. Quando ele estiver aceso, indicará que a máquina está TRABALHANDO. Quando estiver apagado, a máquina poderá estar DISPONÍVEL (caso não haja peça no dispositivo) ou COM ATIVIDADE ACABADA (se houver peça no dispositivo); neste último caso a máquina já acabou de executar uma determinada atividade (usinagem, inspeção, etc) mas não está disponivel pois a peça ainda não foi retirada da máquina. Quando o *led* estiver piscando, então a máquina está QUEBRADA ou COM PROBLEMAS.

Outras duas características importantes na modelagem do comportamento de uma máquina são que o usuário pode: (a) atribuir um valor de probabilidade para que a máquina quebre (estado 4) e (b) pode definir o tempo que a máquina deverá permanecer trabalhando (estado 2).

Isso é importante porque no caso do gerente FMC1, por exemplo, considera-se, por simplificação, que nenhuma das máquinas têm chance de quebrar. Neste caso deve-se atribuir às variáveis *'probabilidades de quebra'* das máquinas o valor zero, que é o mesmo que dizer que existe 0% de chance para que a máquina quebre.

Como serão produzidas três tipos de peças rotacionais diferentes, cada uma delas deve ter seu tempo de usinagem específico. Neste caso então, o *software* de simulação do torno tem que permitir que os tempos de usinagem das peças sejam diferentes.

Como ilustração, considera-se que um aplicativo qualquer deseja utilizar o *software* do dispositivo_Máquina. Supondo-se que a máquina a ser simulada seja uma fresadora, as principais funções a serem utilizadas serão do tipo:

```
[para declarar uma fresadora:]
MÁQUINA   fresadora;

[para inicializar a fresadora, informando seu nome, a operação a ser executada
(p.ex., desbaste, acabamento, ou mais genericamente: produção de uma peça) e a
probabilidade da máquina quebrar.]
        fresadora.cria("nome_da_fresadora","operação_que_executa",prob_quebra);

[para iniciar a operação.  A partir deste ponto, se a máquina não entrar no
estado (4), ela ficará trabalhando durante o tempo de usinagem da peça.]
        fresadora.trabalha("nome_da_peça", tempo_de_usinagem_dessa_peça)
```

5.2 - Os componentes da FMC1

A célula de usinagem foi formada pela integração dos seguintes componentes:

- 4 *buffers* de entrada;
- 1 *buffer* refugo;
- 1 dispositivo_Torno;
- 1 dispositivo_Centro;
- 1 dispositivo_SV;
- 1 robô Rhino XR4;
- 1 controlador MARK-IV; e
- 1 microcomputador.

5.2.1 - O *buffer* de entrada

Para carregar a célula com a matéria-prima, são utilizados *buffers* (chamados também de depósitos, magazines ou *dispensers*) de entrada.

Como são produzidos quatro tipos de peças diferentes, são necessários quatro *buffers* de entrada, cada um deles destinado exclusivamente à entrada de um tipo de peça bruta - PB.

Os *buffers* de entrada são integrados ao gerente da célula através de sensores elétricos. Quando existe pelo menos uma peça num determinado *buffer*, um sinal é enviado ao gerente, informando-o da disponibilidade de tal peça. Esses sinais são muito importantes para a perfeita integração da célula, pois o robô somente deverá ir pegar uma peça quando a mesma estiver no *buffer*.

Caso não haja PB num *buffer* de entrada, o gerente termina de executar as tarefas que ainda são possíveis de serem feitas e logo em seguida interrompe suas atividades, à espera da chegada de PB.

Os quatro *buffers* disponíveis na célula são praticamente iguais, com exceção das suas dimensões e de alguns tipos de sensores.

Conforme pode ser observado na figura 5.6, o *buffer* foi projetado de maneira que, quando uma PB for retirada, as outras PBs escorregam, preenchendo o lugar "vazio" com uma nova peça, exatamente no mesmo ponto onde estava a peça anterior. Isso é muito importante em sistemas FMC, pois em geral o robô não possui capacidade para corrigir distorções de posição.

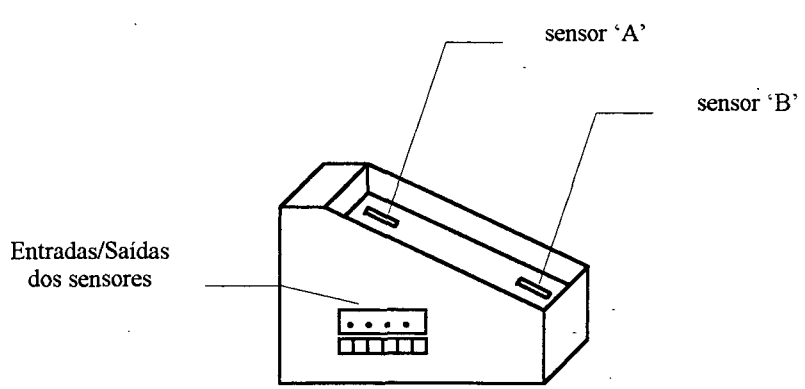


Figura 5.6 - *Buffer* de entrada

A figura 5.7 mostra os quatro *buffers* de entrada utilizados na FMC1. Da direita para esquerda são armazenadas as peças 1,2,3 e 4 respectivamente.

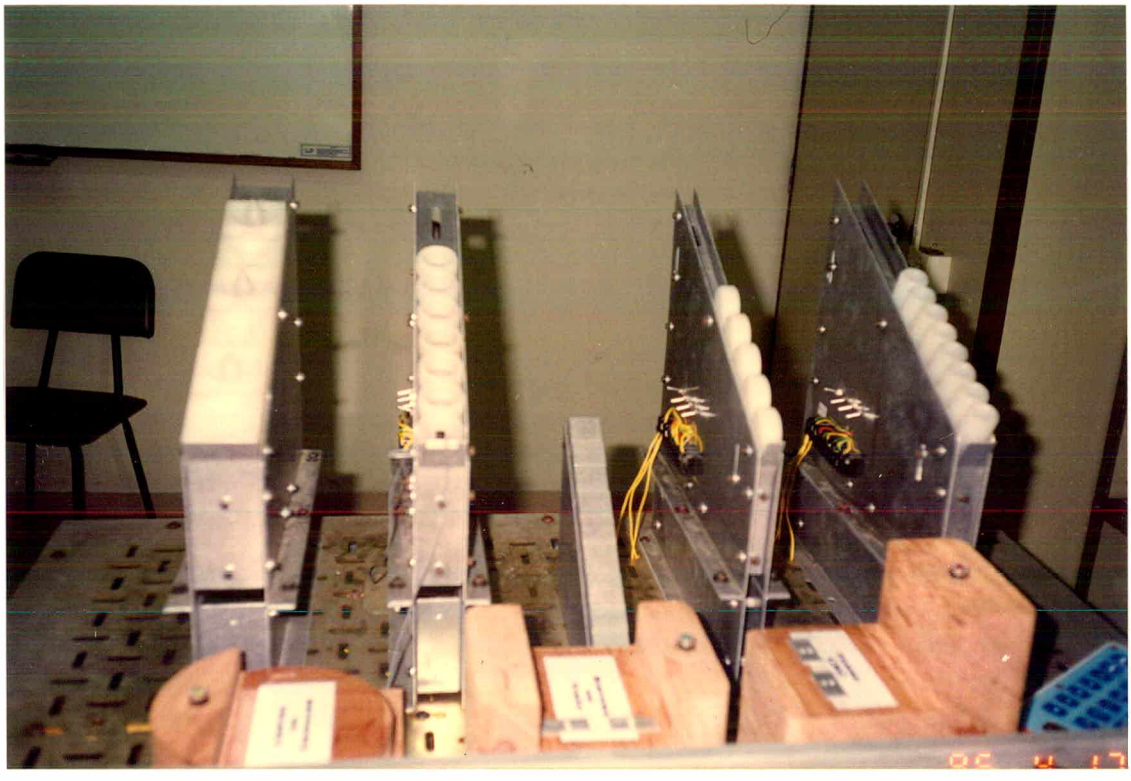


Figura 5.7 - Foto dos *buffers* de entrada da FMC1

5.2.2 - O *BUFFER* REFUGO

O *buffer* refugo é um lugar específico na FMC1 onde o robô deposita as peças consideradas do tipo REFUGO.

5.2.3 - O DISPOSITIVO_TORNO

Na FMC1, o torno CNC é simulado por um dispositivo_Máquina, denominado *dispositivo_Torno*.

O *hardware* do dispositivo_Torno está desenhado na figura 5.8.

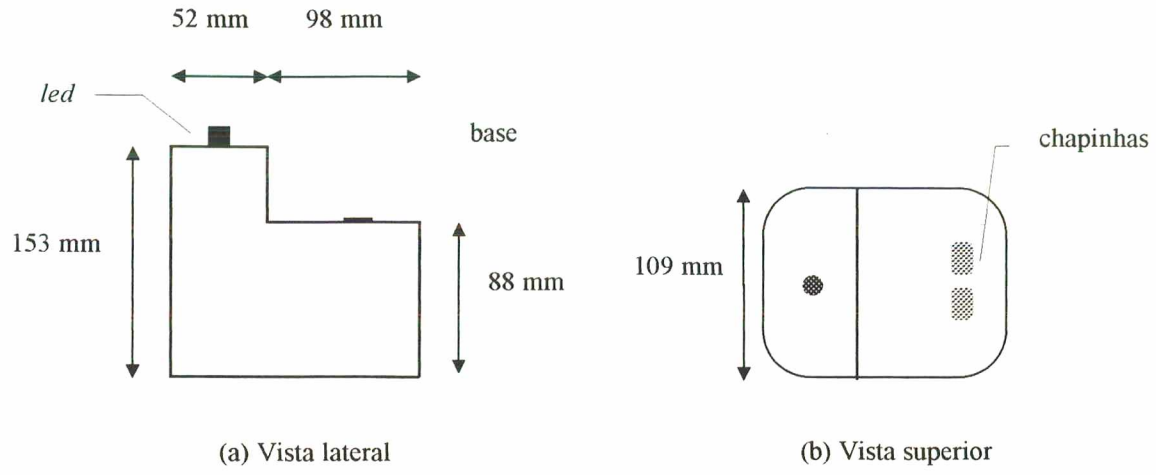


Figura 5.8 - Dispositivo_Torno

Sempre que o robô depositava a peça 2 no dispositivo, devido à sua geometria ela se movia e saía do ponto determinado, pois a superfície cilíndrica da peça rolava sobre a superfície plana do dispositivo. Isto fazia com que o robô não pudesse pegá-la corretamente, ou mesmo, nem conseguia pegá-la.

Colocou-se então duas pequenas chapas feitas em alumínio (chapinhas) para fazer com que a peça, quando depositada entre elas, não pudesse se deslocar.

Um sinal vindo do gerente FMC1 informa ao operador da célula o estado atual do dispositivo. Assim, quando existe este sinal, o *led* do dispositivo_Torno está aceso, indicando que a peça está sendo usinada (estado 2). Quando o *led* está apagado (sem sinal vindo do gerente), ele poderá estar livre (estado 1) ou com uma peça pronta, já usinada (estado 3). O *led* nunca estará piscando pois, como visto anteriormente, considera-se que não haverá quebras ou problemas em nenhuma máquina.

5.2.4 - O DISPOSITIVO_CENTRO

O centro de usinagem da FMC1 é simulado pelo *dispositivo_Centro*, fisicamente representado pelo dispositivo mostrado na figura 5.8 .

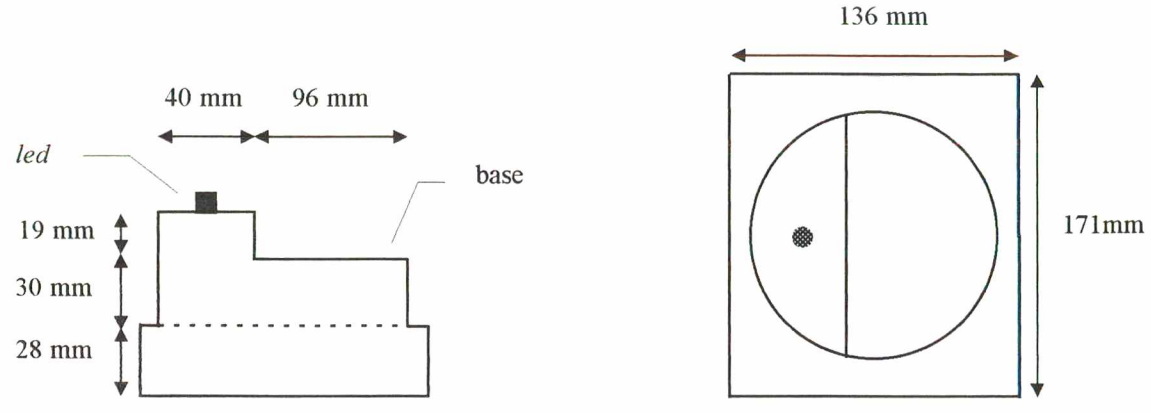


Figura 5.9 - Dispositivo_Centro

Da mesma forma que no dispositivo_torno, um sinal vindo do gerente da célula informa o estado atual do dispositivo_Centro. Assim, quando existe este sinal, o *led* do dispositivo está aceso, mostrando que a máquina centro de usinagem está usinando uma peça.

5.2.5 - O DISPOSITIVO_SV

A máquina que faz a inspeção das peças fabricadas é um sistema de visão, simulado no trabalho *pelo dispositivo_SV* (figura 5.10).

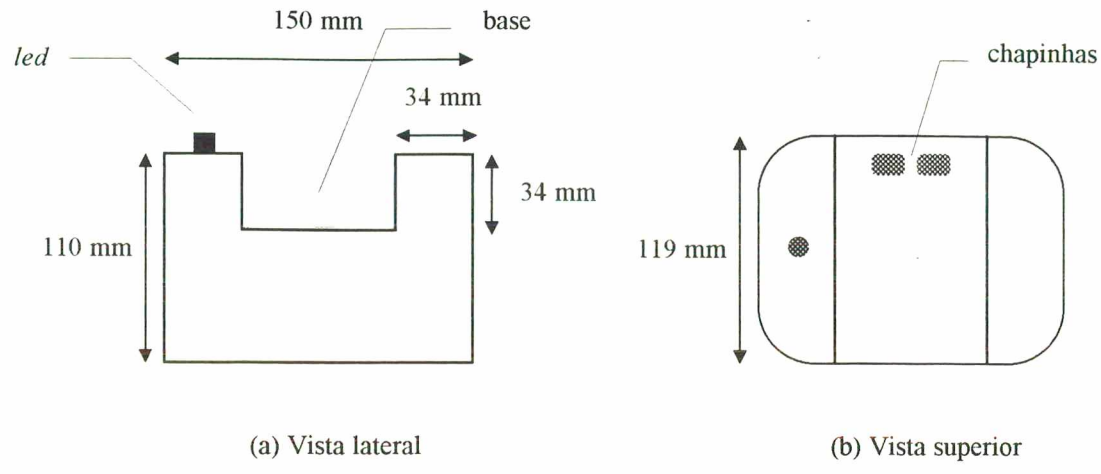


Figura 5.10 - Dispositivo_SV

Quando o *led* do dispositivo_SV estiver acesso, indicará que o sistema de visão está inspecionando uma peça, caso contrário, o dispositivo está disponível ou a peça inspecionada ainda não foi retirada. Também aqui o sinal de alimentação do *led* vem do gerente da célula.

Para as peças rotacionais, o tempo de inspeção é curto (alguns poucos segundos), enquanto que para a peça prismática ele é maior, pois na realidade seria exigido uma quantidade maior de pontos de análise, devido à sua geometria mais complexa.

Por meio da geração de um número aleatório, uma peça será considerada BOA ou RUIM após a sua inspeção. Na configuração da FMC1 proposta, existe 2.5% de chance para que uma peça seja considerada RUIM e, como não podem haver peças RETRABALHO, 97.5% para ela ser considerada BOA.

Essa probabilidade, junto com os tempos de usinagem e inspeção das peças, pode ser facilmente alterado pelo usuário da célula, para isto basta apenas alterar o arquivo que contém a rede de Petri (RdP) da célula (FMC1.RDP) - conforme será visto no ítem 5.4.4.

Como existe apenas um único sistema para inspeção das peças, a RdP da FMC1 deve coordenar as ações e os movimentos do robô de tal forma que apenas uma peça (e tem que ser a peça da vez) utilize o dispositivo_SV num determinado instante.

A figura 5.11 mostra os "dispositivos_Máquina" desenvolvidos.

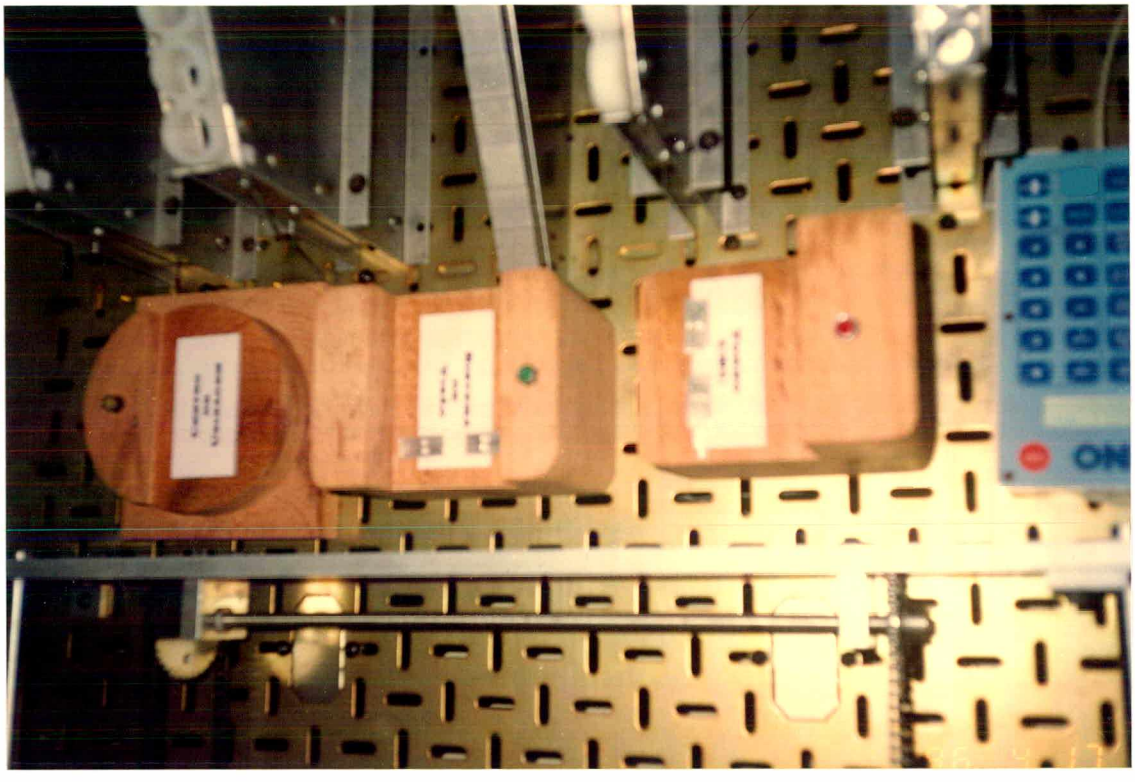


Figura 5.11 - Foto dos dispositivos_Máquina da FMC1

5.2.6 - O COMPUTADOR CENTRAL DA FMC1

É o computador central da célula (CCC) que faz a integração dos dispositivos de uma FMC. É nesse computador que executa o *software* de gerenciamento da célula (gerente FMC), responsável pelo gerenciamento das atividades que ocorrem na célula e também pelo atendimento ao usuário.

Em sistemas FMC industriais, ele interliga os elementos da célula através de uma *local area network (LAN)*, em geral com protocolo de comunicação *field bus*. Neste trabalho entretanto, essa integração física é feita por meio do controlador do robô, que é conectado ao CCC.

A comunicação do computador central com o MARK-IV é estabelecida por meio de um cabo de comunicação serial assíncrono RS232-C, através dos parâmetros de comunicação definidos na seção 4.2.1.

O CCC de fabricação é do tipo 486, DX2, com 66 MHz de velocidade, com disco rígido de 420 MBytes e memória de 4MBytes.

5.3 - INTEGRAÇÃO DOS COMPONENTES

Neste ponto, serão tratadas algumas questões relativas à integração dos componentes da célula. Não serão considerados nesta seção aspectos relativos à programação e utilização, apenas a parte de *hardware* será abordada. Alguns problemas encontrados, junto com as soluções elaboradas, também serão descritos.

Essa descrição será feita obedecendo-se a mesma sequência das etapas executadas na prática, assim o leitor poderá analisar melhor o desenvolvimento do trabalho.

5.3.1 - AS CONEXÕES NO MARK-IV

A primeira etapa a ser desenvolvida foi configurar as entradas e saídas do controlador do XR4. Nesta aplicação, configurar o MARK-IV significa: (a)alimentar as portas de entrada, (b)aterrar as portas de saída, (c)definir quais entradas/saídas serão utilizadas, e (d)verificar quais dispositivos serão integrados e como será estabelecida esta integração.

Algumas das conexões já haviam sido feitas por bolsistas do Laboratório de Automação Industrial (LAI), mas tiveram que ser refeitas pois estavam operando de maneira incompatível com o que era necessário para esta aplicação.

A figura 5.12 mostra como foi configurado o controlador MARK-IV após a integração dos componentes da FMC1.

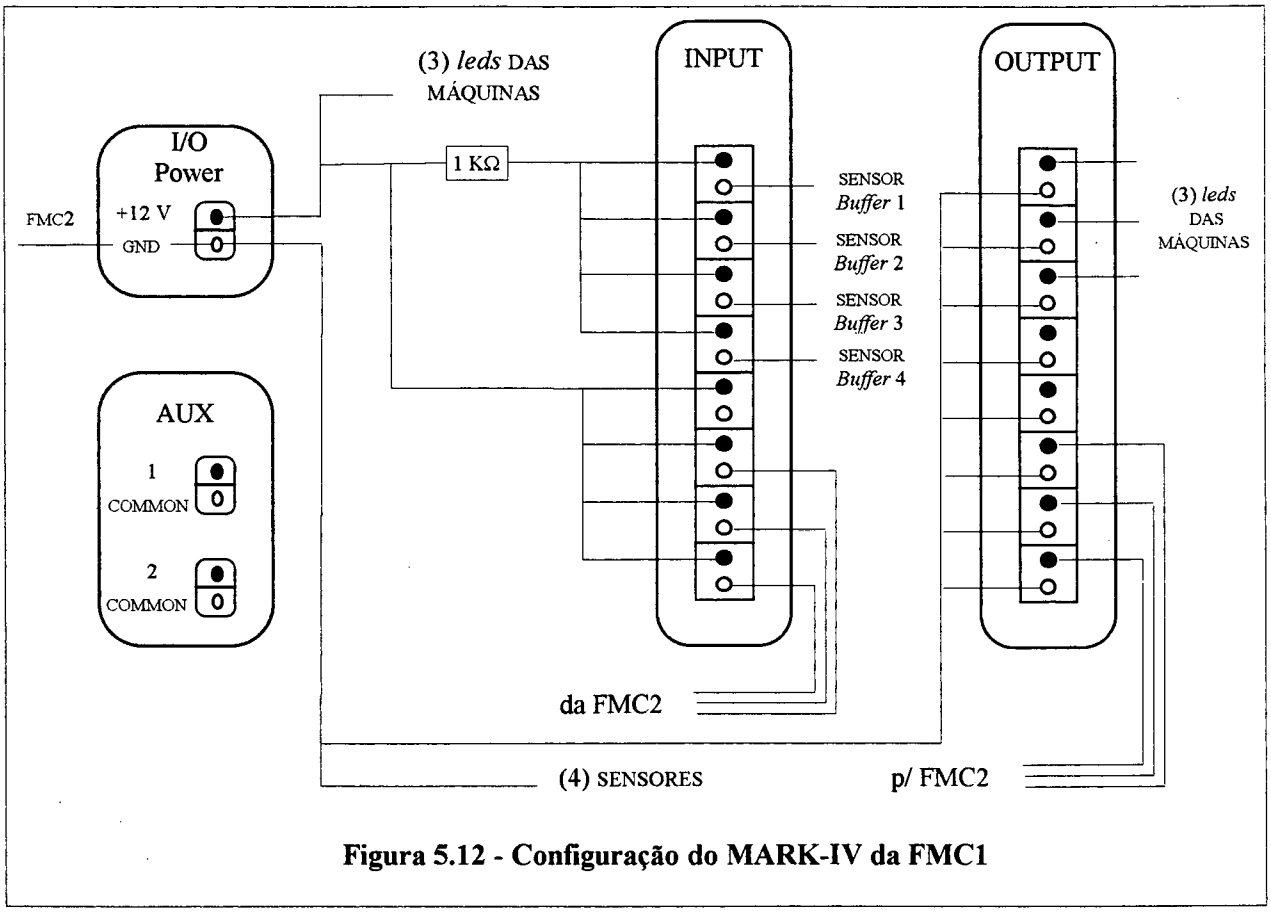


Figura 5.12 - Configuração do MARK-IV da FMC1

5.3.2 - PROBLEMAS E SOLUÇÕES

Alguns problemas ocorreram durante a implementação do trabalho, provando que prática e teoria nem sempre andam em perfeita harmonia. Um deles foi que o *driver* de comunicação com o MARK-IV, por exemplo, que deveria apenas enviar e receber

bytes, teve em seu desenvolvimento vários pontos de difícil solução. O próprio Presidente da empresa *Rhino Robots* disse que "algumas vezes é muito difícil escrever uma boa rotina de comunicação, por isso esta é uma área em que se deve prestar muita atenção, e se certificar que a confiabilidade no código está alta".

Alguns dos problemas encontrados no decorrer do desenvolvimento do trabalho e as soluções adotadas já foram descritos anteriormente, entretanto outros devem ser também mencionados:

(a) As peças, por serem feitas de NYLON, possuem um baixo peso, e isto fez com que alguns sensores na posição 'A' nos *buffers* de entrada (figura 5.6) impedissem as peças de deslizar. Esses sensores foram então retirados do *buffer*.

(b) Devido à incompatibilidade das conexões existentes inicialmente no MARK-IV, quando determinadas entradas estavam ativas (ON), outras eram automaticamente desativadas (OFF). Descobriu-se depois que havia necessidade de se separar os sinais utilizados pelos sensores dos sinais de integração, os quais são utilizados para integrar as duas células (capítulo 7). Esta divisão foi então estabelecida através da diferença de corrente/tensão utilizada. Dessa forma, os sinais utilizados pelos sensores foram alimentadas com baixa corrente e tensão, enquanto que aqueles feitos para a integração das células utilizaram um valor maior de corrente/tensão. Isto pode ser observado pela não colocação do resistor de 1kΩ na entrada dos sinais de integração figura 5.12.

(c) Devido ao baixo peso da peça do tipo (3), o sensor do *buffer* de entrada 3 (i.e., o sensor na posição 'B') não podia ser ativado. Desenvolveu-se então um sensor próprio (figura

5.13), que apesar de bastante simples, é muito importante para o funcionamento da célula. Sua operação segue a mesma idéia dos sensores industriais, baseando-se no fechamento de um circuito por onde pode fluir uma corrente elétrica.

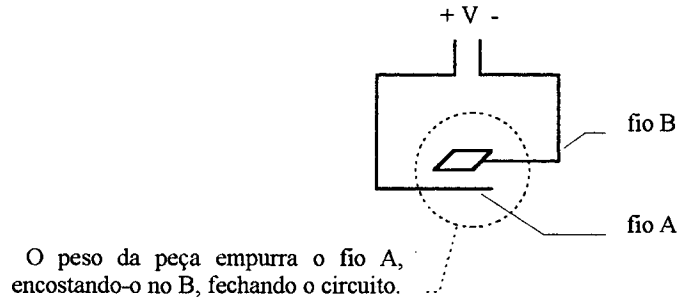


Figura 5.13 - O sensor do *buffer* de entrada 3

(d) As peças (3) e (4) não podiam ser pegas pelo robô, pois a abertura máxima da sua garra é de aproximadamente 30 mm, enquanto que essas peças possuem 31 e 44 mm de expessura respectivamente. Foi então desenvolvido um *ampliador* para a garra do robô (figura 5.14). Uma outra solução seria comprar uma nova garra, mas esta não foi considerada pois a solução do ampliador era mais rápida para ser executada e sem custo adicional.

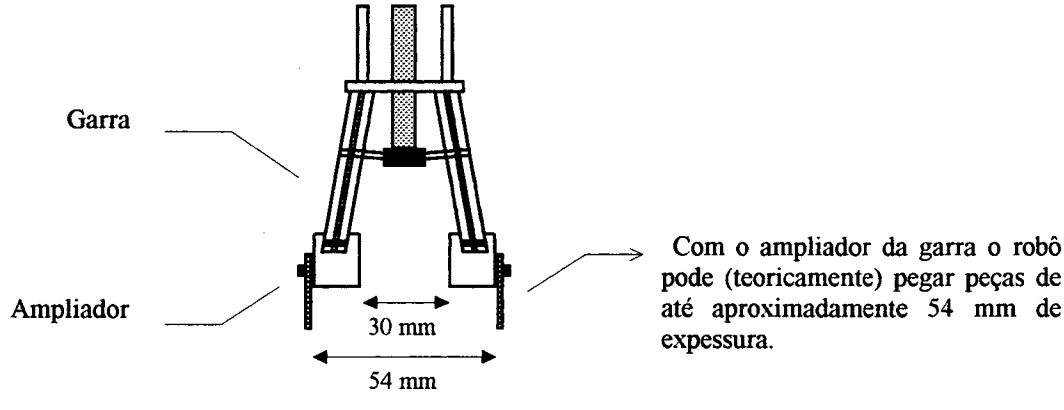


Figura 5.14 - O ampliador da garra do robô

(e) A baixa precisão e repetibilidade do XR4 são sem dúvida os problemas mais difíceis de serem tratados, haja visto que são causados pelas características de fabricação do robô. Não se consegue, por exemplo, inicializar o robô numa posição sempre constante, conhecida. O leitor mais familiar com programação de robôs poderia perguntar: por quê não utilizar o seu *hard home* (HH)? Como os robôs utilizados neste trabalho são acadêmicos, mesmo os seus HHs têm imprecisões. Cada vez que se manda o robô para o HH, ele pára numa posição diferente. Apesar dessa imprecisão ser pouco perceptível, ela pode afetar negativamente toda a sequência de execução dos próximos movimentos do robô, pois todos os pontos ensinados (mais de cento e vinte na FMC1) são referenciados ao HH do robô.

Um outro problema surge quando o robô tem que repetir um mesmo ciclo várias vezes. Existe um erro que se acumula na medida em que o robô repete um ciclo. No caso de sistemas de manufatura, em particular em sistemas FMC, um robô com este tipo de limitação jamais poderia se utilizado, pois um sistema como esse deve operar durante horas consecutivas, repetindo um mesmo ciclo.

Não existiu uma solução concreta para estes dois problemas, haja visto que eles são causados pelas características de construção do robô, as quais não podem ser alteradas. Entretanto, descobriu-se meios para fazer com que esses efeitos fossem diminuídos.

Para o primeiro problema (baixa precisão), a solução adotada foi fazer com que antes de se iniciar a operação da célula, o robô fosse para um ponto pré-determinado, onde então o responsável pela célula (operador) pode verificar se aquela posição corresponde à posição onde o robô realmente deveria estar (*"ponto de confirmação"*). Se o robô estiver na posição correta, o ciclo pode então ser iniciado. Porém, se a posição não for satisfatória o robô deverá ser reinicializado. Este procedimento deve ser repetido até que o robô tenha sido inicializado na

posição desejada. Na prática, geralmente duas ou três tentativas no máximo já são suficientes.

Com relação ao segundo problema (baixa repetibilidade), descobriu-se que diminuindo a aceleração e a velocidade dos motores, diminui-se os efeitos desse problema. Entretanto, deve-se buscar uma relação ótima entre repetibilidade e aceleração/velocidade, pois percebe-se que melhorando-se bastante um fator, prejudica-se na mesma proporção o outro. Descobriu-se também que existem pontos críticos, isto é, pontos que exigem uma maior precisão e repetibilidade. Esses pontos são aqueles onde o robô deve aproximar a garra para pegar ou depositar uma peça. Nesses pontos, a aceleração e velocidade devem ser colocadas em torno de 30% à 40% da aceleração e velocidade máximas do sistema. Nos outros pontos, consegue-se trabalhar satisfatoriamente com cerca de 75% à 85%.

5.4 - O GERENTE FMC1

O *software* de gerenciamento da célula flexível de fabricação (gerente FMC1) é o programa que roda no computador central da célula. Ele é responsável pelo gerenciamento das atividades que ocorrem na célula e pelo atendimento ao usuário, que poderá ser o operador responsável pela célula e/ou um gerente FMS. Neste trabalho, a figura do gerente FMS, localizado num nível hierárquico acima do nível onde se encontram as FMCs, não é considerada, e portanto, o gerente da célula atende somente ao operador da FMC.

As atribuições que o gerente FMC1 tem que executar estão descritas detalhadamente a seguir. Nesta descrição, faz-se uma separação entre funções de atendimento à célula (FAC) e funções de atendimento ao usuário (FAU).

As FAC são externas ao gerente e são definidas pelo usuário através da rede de Petri da célula. As FAU, por sua vez,

são consideradas internas, pois não podem ser alteradas pelo usuário, podendo apenas ser acessadas através das opções nos menus da interface gráfica do gerente.

O gerente deve procurar atender tanto as FAC quanto as FAU. Contudo, como poderá ser observado a seguir, estas atividades não são executadas simultaneamente, mas sim através de um pseudo-paralelismo.

5.4.1 - As FAC do gerente FMC1

As funções de atendimento à célula de fabricação são modeladas pelo usuário da célula através de uma rede de Petri interpretada (RPI). Caso o usuário queira mudar a atual configuração da FMC1, ele poderá fazê-lo através da especificação de uma nova RPI.

O gerente permite que sejam produzidas peças diferentes, com etapas de produção e, como será visto no próximo capítulo, com ciclos de montagem diferentes.

Essa rede de Petri deve ser escrita sob a forma de um arquivo texto chamado: "FMC1.RDP". Se o leitor desejar, pode recorrer ao anexo III para verificar como foi feita a descrição da RdP da FMC1.

As principais FAC do gerente FMC1 são:

1. Comandar/controlar os componentes da FMC1 (os dispositivos_Máquina, o robô, etc).

2. Coordenar a sequência dos movimentos do robô e as operações nas máquinas.

3. Estabelecer a integração (lógica) e o sincronismo com a célula de montagem (capítulo 7).

4. Permitir a alteração da configuração da célula, quando, por exemplo, novos tipos de peças tiverem que ser produzidos. Neste caso o gerente deverá permitir que se entre com uma nova RdP e com novos pontos ensinados (FAU).

5.4.2 - As FAU do gerente FMC1

Uma série de funções podem ser acessadas pelo usuário através das opções nos menus. A seguir é feito uma rápida descrição dessas funções, sendo que as frases em destaque (em negrito e itálico) mostram como elas são encontradas nos menus, facilitando-se a compreensão da seção 5.4.5, que mostra a estrutura de navegação do gerente.

1. Inicializar a célula (***Iniciar FMC***):

Antes de iniciar a operação na célula, o usuário deve configurá-la. Com este procedimento, o MARK-IV e o robô são inicializados (*hard home*, valores das velocidades e acelerações dos motores, etc), o gerente carrega a RPI e faz a leitura dos pontos ensinados.

O usuário pode inicializar a célula com valores pré-definidos (***defaults***) ou pode ele mesmo selecionar os parâmetros (***via arquivo***), através de um arquivo texto com nome "FMC1.CFG".

2. Iniciar ou reiniciar a operação na célula (***(Re)Iniciar operação***):

Após inicializar a célula, pode-se dar início à execução das operações na célula. Antes disto entretanto, o gerente pergunta ao usuário se ele aceita o *ponto de confirmação* mostrado (seção 5.3.2). Se o usuário não aceitar o ponto, a FMC deve ser inicializada novamente.

Esta opção serve para reinicializar a célula em situações como as descritas acima, e também quando o usuário decide interromper (sem abortar) a execução (***parar operação***).

3. O usuário pode verificar qual o estado dos ***sensores***, das ***máquinas***, do controlador ***MARK-IV*** e dos sinais

da/para célula de montagem (***FMC2***) a qualquer momento, através da opção: ***Verificar status***.

4. Com o objetivo de acompanhar o andamento da produção da célula, o usuário pode saber o número de peças boas e ruins fabricadas até aquele momento, junto com informação sobre a taxa de produção da célula e de utilização das máquinas e do robô (***Produção***).

5. Um relatório de acompanhamento geral da FMC, junto com uma descrição de produtividade e de *status* pode ser gerado sob a forma de um arquivo (***Criar relatório***) - "FMC1.REL". Fora do ambiente do gerente o usuário poderá então imprimi-lo.

6. O usuário pode comandar o robô, sem precisar inicializar toda célula. Através desta opção (***Robô***), o gerente permite que o usuário: (a) veja a configuração do robô (**Mostrar configuração**), (b) mova, a partir do teclado, os seus motores, mostrando sempre em que posição o motor está (**Motores**), (c) ensine os pontos ao robô (**Ensinar pontos**), (d) leia pontos já ensinados (**Ler pontos**) e (e) mova o robô para um ponto ensinado qualquer (**Ir para**).

7. Uma tela de ajuda (***Help***) permite que um novo usuário possa trabalhar com a FMC1, verificando, por exemplo, se as posições dos equipamentos não foram modificadas, se a porta de comunicação está estabelecida corretamente, etc.

8. O usuário pode guardar os pontos que ensinou ao robô (***salvar pontos***). Esses pontos ensinados ficam armazenados num arquivo chamado "FMC1.PTS". O usuário também pode salvar a atual configuração do sistema (***salvar configuração***) no arquivo "FMC1.CFG".

9. O gerente permite que o usuário possa parar a execução das atividades na célula (***parar operação***), podendo depois reinicializar as atividades no ponto que elas foram interrompidas. Esta opção pode ser usada, por exemplo, para executar passo-a-passo as atividades na célula.

Em situações de pane ou emergência, o usuário deverá pressionar a *"tecla de parada de emergência"* (tecla 'a') que interrompe naquele instante as atividades e movimentos sendo executados na célula. As diferenças entre a opção ***parar operação*** e a ***parada de emergência*** são que esta, após ativada, requer uma nova inicialização da célula, enquanto que após aquela basta apenas selecionar a opção ***(Re)Iniciar;*** uma outra diferença está no fato que a parada de emergência é executada quase que instantaneamente, enquanto que a outra pode demorar um pouco mais, pois espera que a atividade em execução termine.

5.4.3 - A ESTRUTURA DE EXECUÇÃO DO GERENTE FMC1

A figura 5.15 mostra como foi desenvolvida a estrutura de execução interna do gerente da FMC1. Nessa estrutura, pode-se observar que as FAC e as FAU acontecem sequencialmente e não simultaneamente, como desejado. Entretanto, em tempo de execução, isto é, executando-se o ciclo mostrado na figura, elas parecem acontecer paralelamente (pseudo-paralelismo).

Quanto menor for o número de transições na RPI, menos tempo levará para disparar uma transição e consequentemente, mais próximo do paralelismo funcionará o sistema. Caso o número de transições na rede seja elevado, um certo tempo será gasto para procurar a transição a ser disparada e com isso, ficará evidente que as FAC e as FAU não ocorrem simultaneamente.

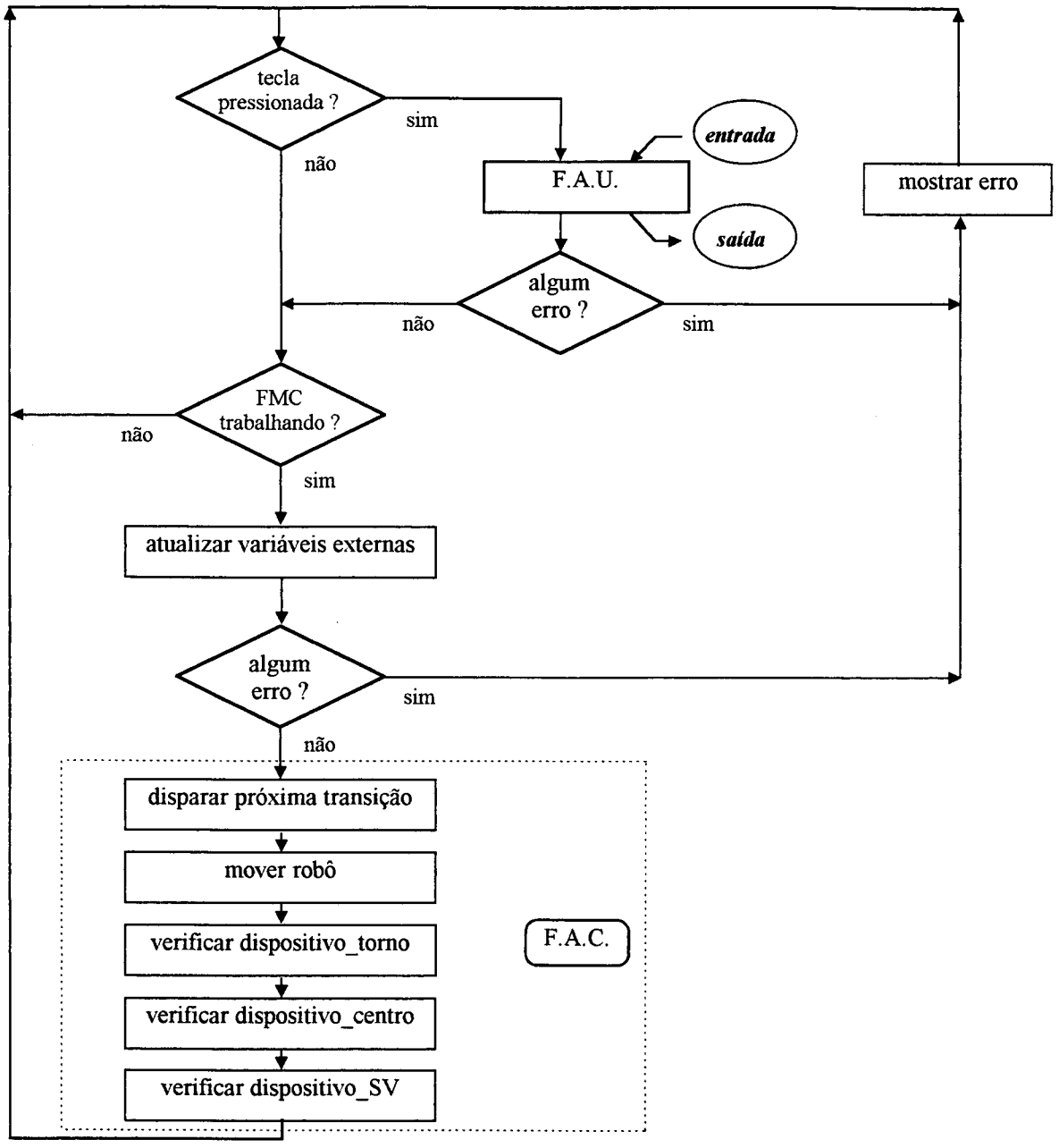


Figura 5.15 - Estrutura interna do gerente FMC1

5.4.4 - AS VARIÁVEIS EXTERNAS

O gerente da célula possui um conjunto de variáveis que são utilizadas tanto pelo seu próprio algorítmo de execução quanto pela rede de Petri especificada pelo usuário da célula.

Essas variáveis são utilizadas para fazer o interfaceamento entre o que o usuário deseja (descrito no arquivo RPI da rede) e os dispositivos da FMC1 conectados ao MARK-IV (ver figura 5.16). Por isso são chamadas de variáveis externas (VE), pois são acessadas tanto internamente pelo gerente quanto externamente pelo usuário.

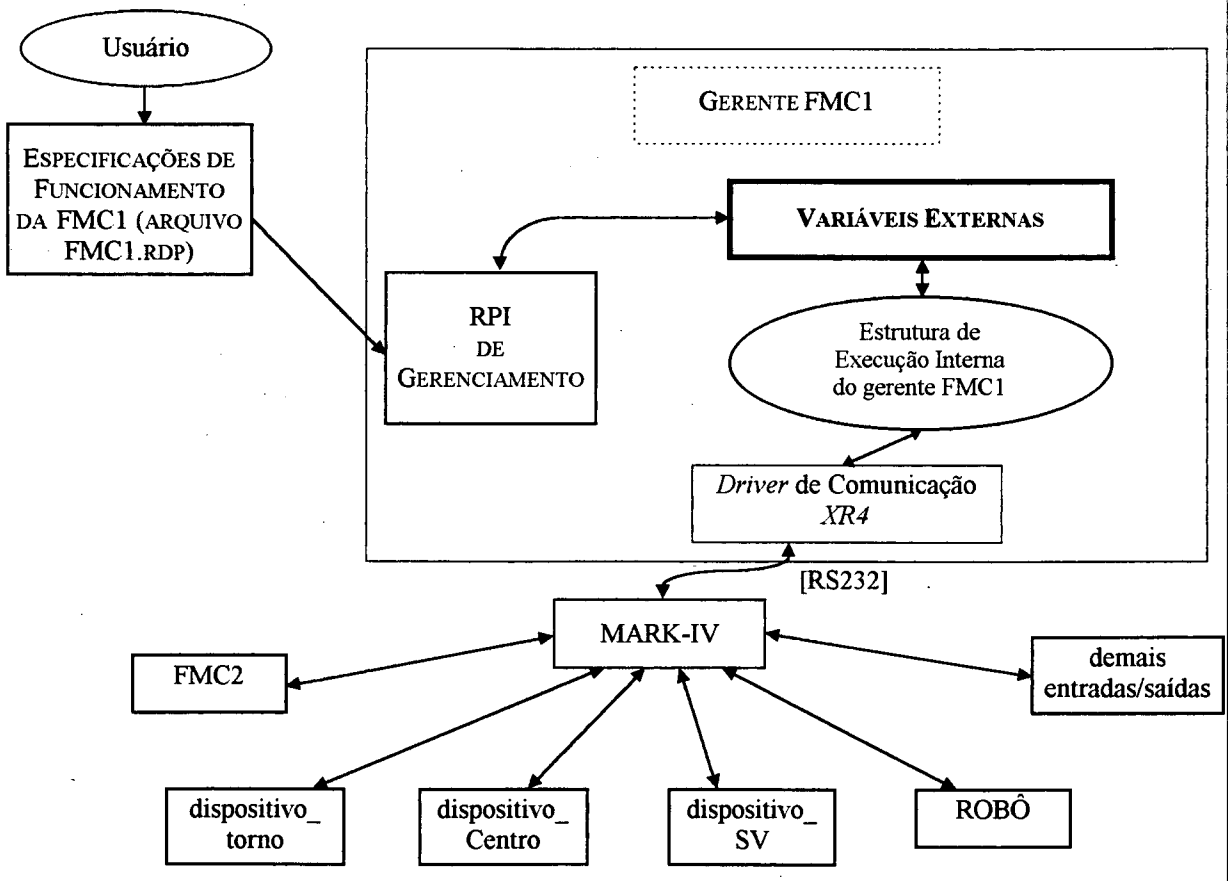


Figura 5.16- A função das variáveis externas

Através das VE, o usuário inicia uma operação na máquina, manda o robô para um ponto ensinado, faz a leitura de uma entrada, etc.

As variáveis externas utilizadas na FMC1 estão rapidamente descritas a seguir.

- **input1, input2 .. input8**: informam o estado das entradas do controlador MARK IV. Por exemplo, se *input1=0* , então significa que a entrada 1 está desativada (OFF), e se *input1=1*, então ela está ativa (ON);

- **output1, output2 .. output8**: informam sobre o estado da saídas do controlador. Da mesma forma, 1 para saída ON e 0 para saída OFF;

- **npb1, npb2, npb3 e npb4**: número de peças(i) que foram consideradas BOAS [i=1,2,3 ou 4].

- **npr1, npr2, npr3, npr4**: número de peças(i) consideradas RUINS.

- **pmr**: informa qual o próximo movimento a ser executado pelo robô.

- **er**: indica qual o estado do robô no momento (*0:robô parado, 1:executando movimento*).

- **gripper**: informa ao robô se ele deve fechar (*gripper=-1*) ou se ele deve abrir (*gripper=1*) a garra (*gripper*).

- **eg**: informa o estado atual da garra do robô (*0:aberto, 1: fechado*).

- **usina1, usina2, usina3, usina4**: sinal utilizado para indicar a usinagem das peças. Eles também são utilizados para informar ao gerente o tempo de usinagem (aproximado) de cada peça. O valor inicial de usina1, por exemplo, declarada na RPI, informa ao gerente o tempo de usinagem da peça 1.

- **inspeciona1,inspeciona2, inspeciona3, inspeciona4**: sinal para indicar a inspeção das peças. Como usina(i), o valor

inicial de inspeciona(i) indica o tempo aproximado de inspeção da peça (i) [i=1,2,3 ou 4].

- **probPeca1, probPeca2, probPeca3, probPeca4:** o usuário deve definir a probabilidade para que uma peça(i) seja considerada BOA ou RUIM.

O valor de probPeca(i) é gerado aleatoriamente no gerente FMC e está sempre compreendido entre 0 e 999. Por isso, se o usuário quer por exemplo que haja 97,5% de chance para que a peça 1 seja considerada BOA, então *probPeça1 >= 25*, pois a probabilidade de ser gerado um número maior ou igual a 25 entre 0 e 999 é de 97,5%.

- **alteraAS, alteraVS:** Foi visto que o robô possui muita imprecisão e pouca repetibilidade. Como já mencionado também, alterando-se a aceleração e a velocidade do robô consegue-se diminuir os efeitos desses erros. Por esta razão o usuário pode definir qual aceleração (*alteraAS*) e qual velocidade (*alteraVS*) ele quer que o robô tenha quando for executar um determinado movimento. *AlteraAS* e *alteraVS* devem estar compreendidos entre 0 e 100, pois correspondem a valores percentuais da aceleração e velocidade máximos do sistema (valores de fabricação). Por exemplo, se alteraVS=30, então a velocidade do robô será setada a 30% da velocidade máxima.

5.4.5 - ESTRUTURA DE NAVEGAÇÃO ATRAVÉS DOS MENUS

Diversas opções são oferecidas ao usuário da FMC1. Essas funções estão organizadas em grupos, representados através de menus, os quais possuem uma relação entre si. O usuário pode então "navegar" através desses menus, conforme mostrado na figura a seguir.

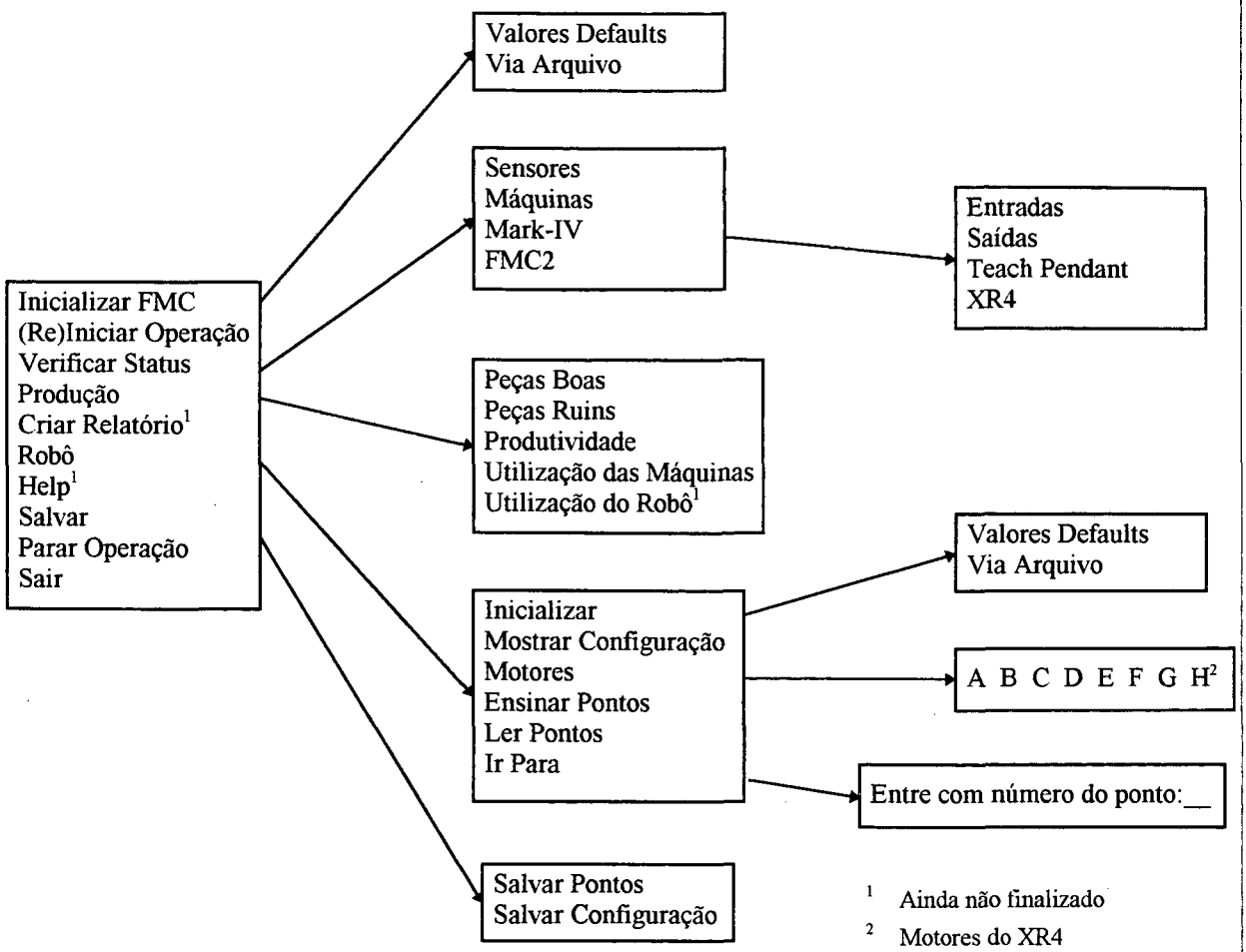


Figura 5.17 - Estrutura dos menus do gerente FMC1

5.5 - A UTILIZAÇÃO DA FMC1

A FMC1 foi desenvolvida para produzir quaisquer tipos de peças dentro da configuração proposta (4 *buffers* de entrada, 1 *buffer* refugo, 1 torno CNC, 1 sistema de visão, 1 centro de usinagem CNC e 1 robô XR4). No presente trabalho entretanto, somente serão produzidos quatro tipos de peças, ilustradas anteriormente nas figuras 5.2 à 5.5.

As considerações sobre a sequência das operações a serem executadas na FMC1 para fabricação dessas peças foram as seguintes:

1°. Uma peça deve ser produzida e, se após a inspeção ela for considerada BOA, deve ser enviada à FMC2 (célula de montagem). Esta peça não pode permanecer na célula pois considera-se que não deve haver estoques intermediários.

2°. O produto final, que sai da FMC2, será formado a partir da montagem das quatro peças produzidas pela FMC1. Por isso, em cada ciclo um tipo diferente de peça deve ser produzido. Não adiantaria, por exemplo, produzir três peças do tipo (2) e nenhuma do tipo (1).

3°. As três peças rotacionais devem ser montadas sobre a peça prismática (peça base), e por isso, esta deve ser a primeira peça a ser fabricada e enviada à celula de montagem.

4°. Para facilitar a montagem na FMC2, deve-se enviar a peça (3) antes da peça (1), e portanto, a ordem definida para a produção das peças foi: produzir peça (4), depois peça (3), peça (2) e então a peça (1). Poderia também ser: 4→2→3→1 ou 4→3→1→2, mas devido ao posicionamento lado-a-lado dos *buffers* de entrada, o ciclo feito pela sequência **4→3→2→1** era o mais rápido, pois o robô precisava se deslocar menos. O gerente controla a sequência das peças que devem ser produzidas através da RPI por meio de uma variável interna chamada *"peça da vez"* (pv). Assim, após produzir a peça 3, por exemplo, pv deverá ser igual a 2.

Como existem duas máquinas, na prática uma peça rotacional é produzida simultaneamente com a peça prismática.

5°. Enquanto uma peça está sendo inspecionada, o robô deve ficar parado. Ele não pode ir pegar uma PB para carregar uma máquina livre porque:

(a) suponha que o torno usinou a peça 3 e que esta seja a peça da vez. O robô então a leva para a máquina de inspeção. Enquanto a peça 3 está sendo inspecionada (lembre-se que o torno está livre), o robô carrega o torno com a peça 2 (sequência de usinagem no torno: 3→2→1→3). Entretanto, se após a inspeção a peça 3 for considerada RUIM, este mesmo tipo de peça deverá ser produzido novamente pelo torno, mas, como foi dito anteriormente, o torno está usinando a peça 2, a qual não poderá ir para a inspeção pois não é a peça da vez (não pode ser enviada à célula de montagem). Para evitar este bloqueio, uma solução seria implementar um *buffer* intermediário. Entretanto, a presença de estoque intermediário foge dos objetivos deste trabalho. Por isto, enquanto uma peça está sendo inspecionada, o robô deve permanecer parado, esperando o fim da atividade. Um outro motivo para esta consideração é que:

(b) como o tempo de inspeção das peças é curto, se comparado com o tempo que o robô levaria para deslocar-se para pegar a peça e depois levá-la até a máquina, é mais vantajoso para a célula que o robô permaneça esperando o fim da inspeção.

Pode-se perceber que a terceira e a quarta considerações já são referentes à integração das FMCs, como será visto no capítulo 7.

5.5.1 - A RPI DESENVOLVIDA

A figura a seguir, dividida em seis páginas, mostra a rede de Petri interpretada utilizada pelo gerente da FMC1 para fabricar as quatro peças definidas neste trabalho. Após o desenho da rede, tem-se uma legenda a qual o leitor poderá consultar para melhor entender o significado das transições, dos lugares e das variáveis da rede.

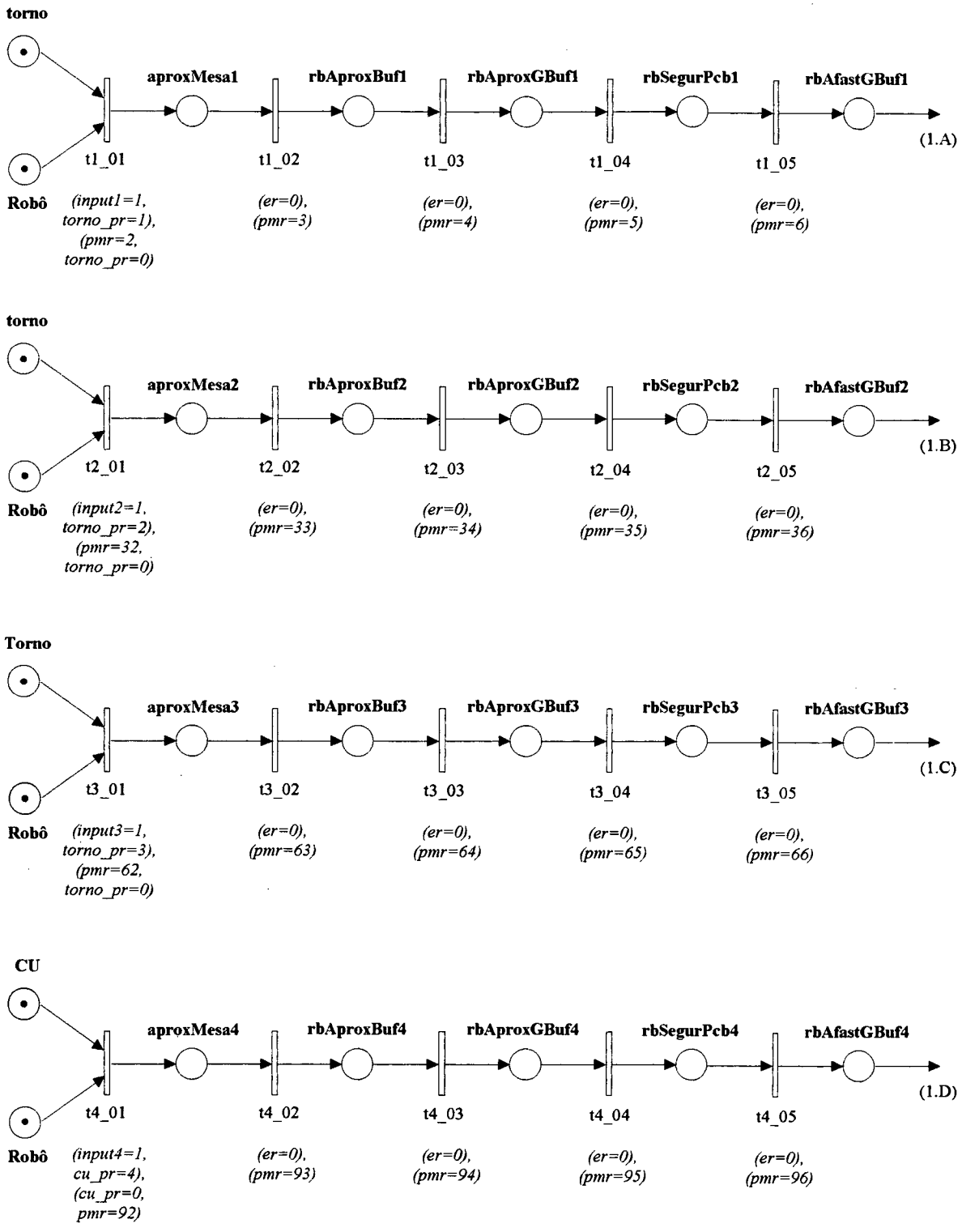


Figura 5.18 - A RPI da FMC1 - parte I

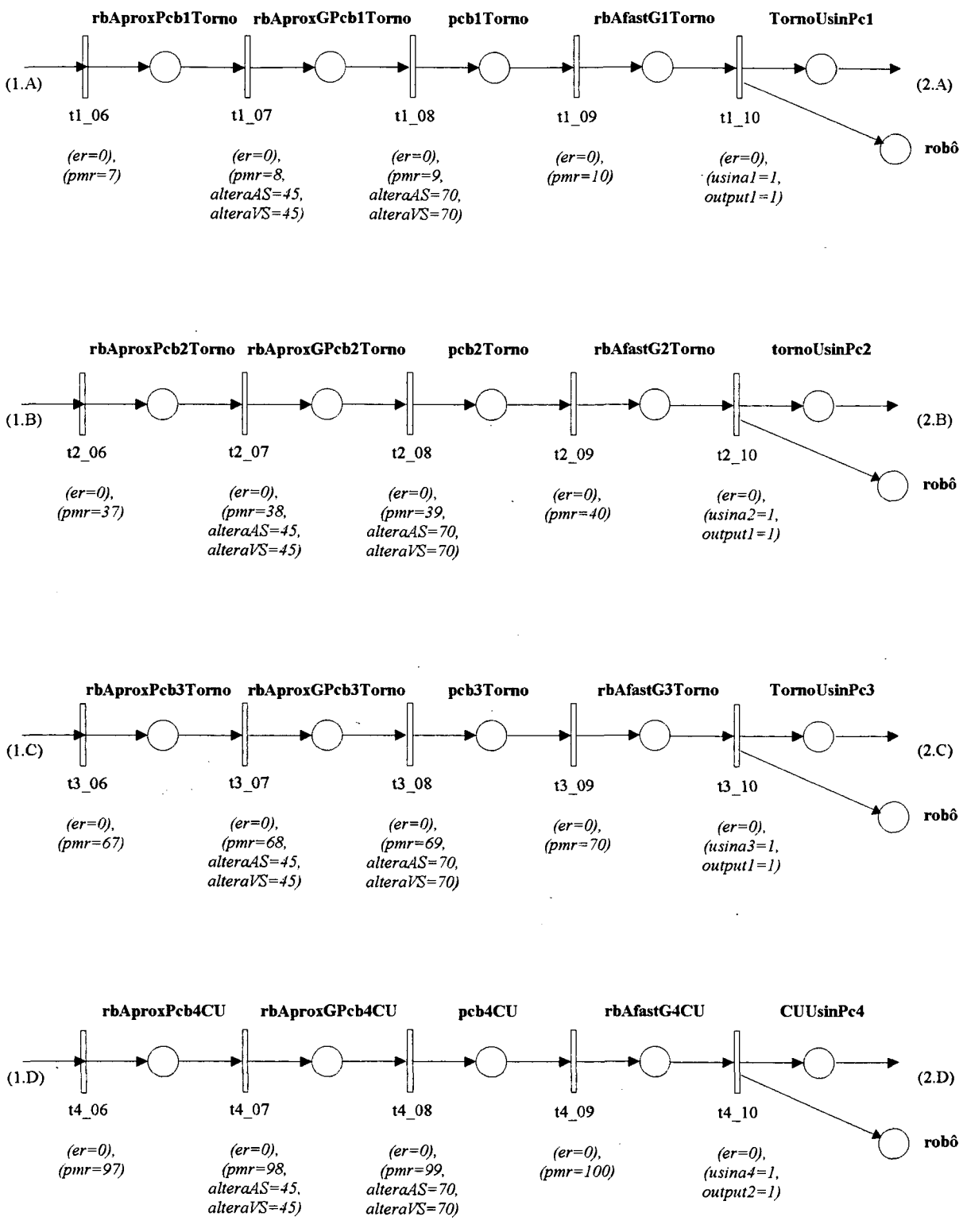


Figura 5.18 - A RPI da FMC1 - parte II

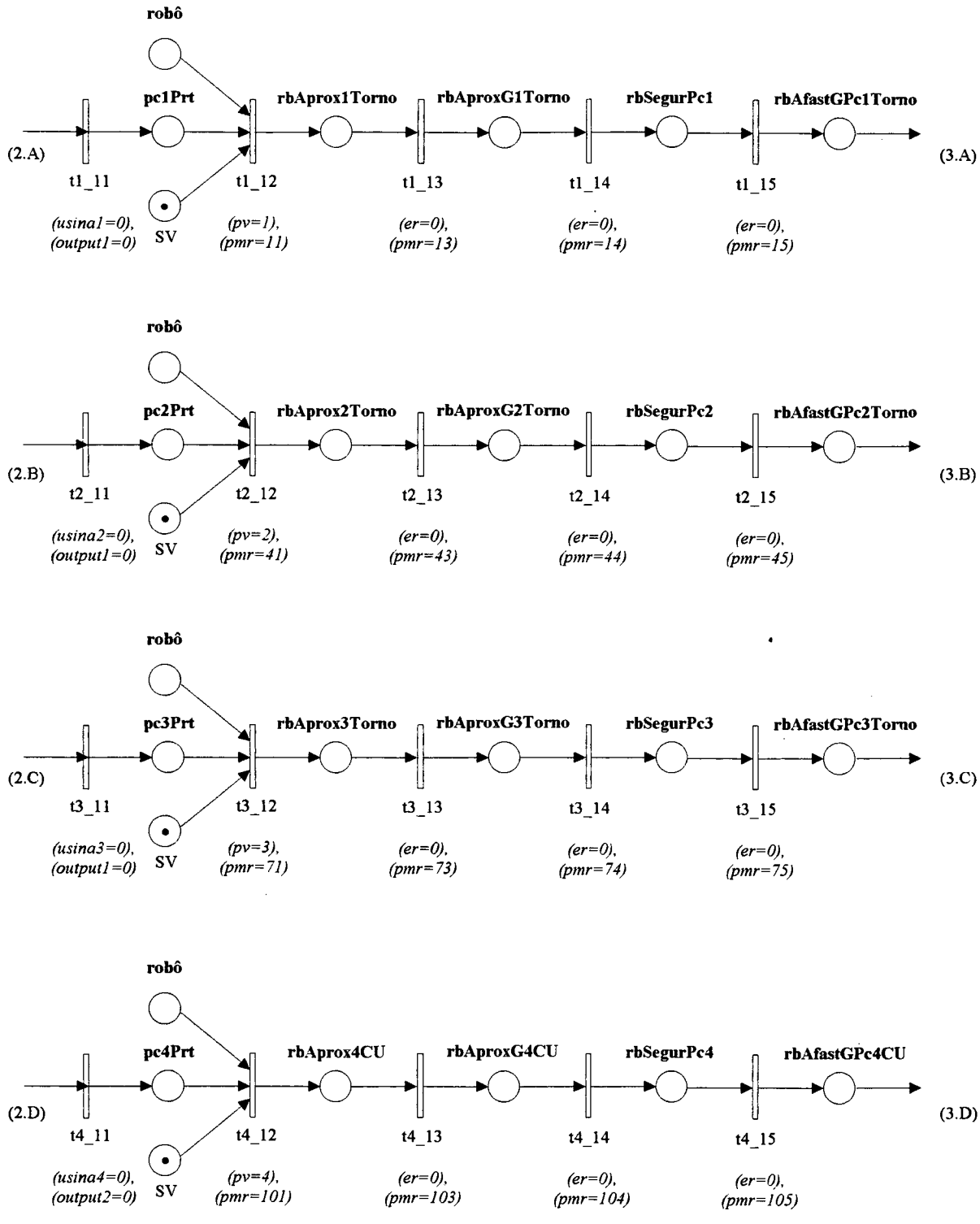


Figura 5.18 - A RPI da FMC1 - parte III

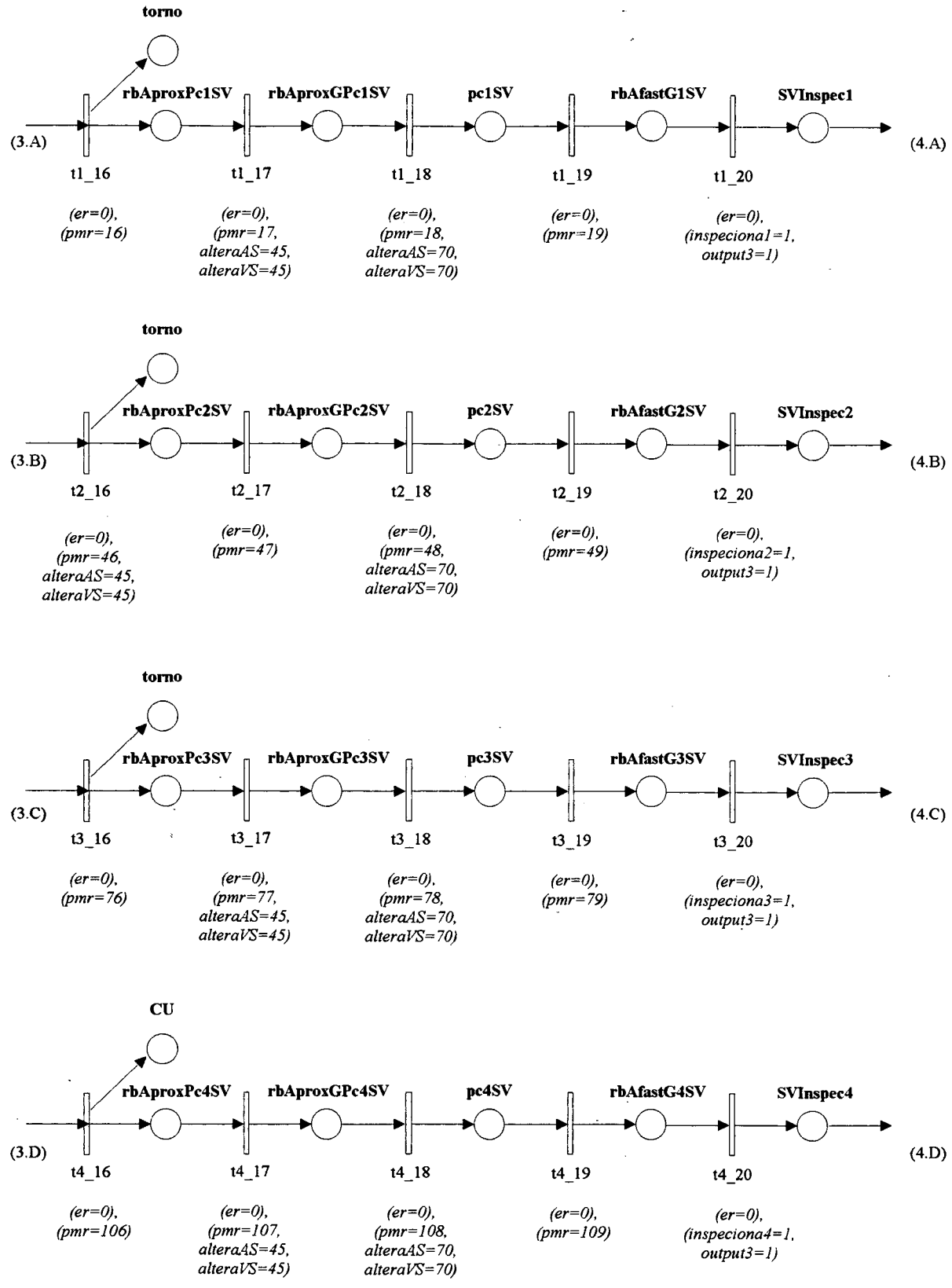


Figura 5.18 - A RPI da FMC1 - parte IV

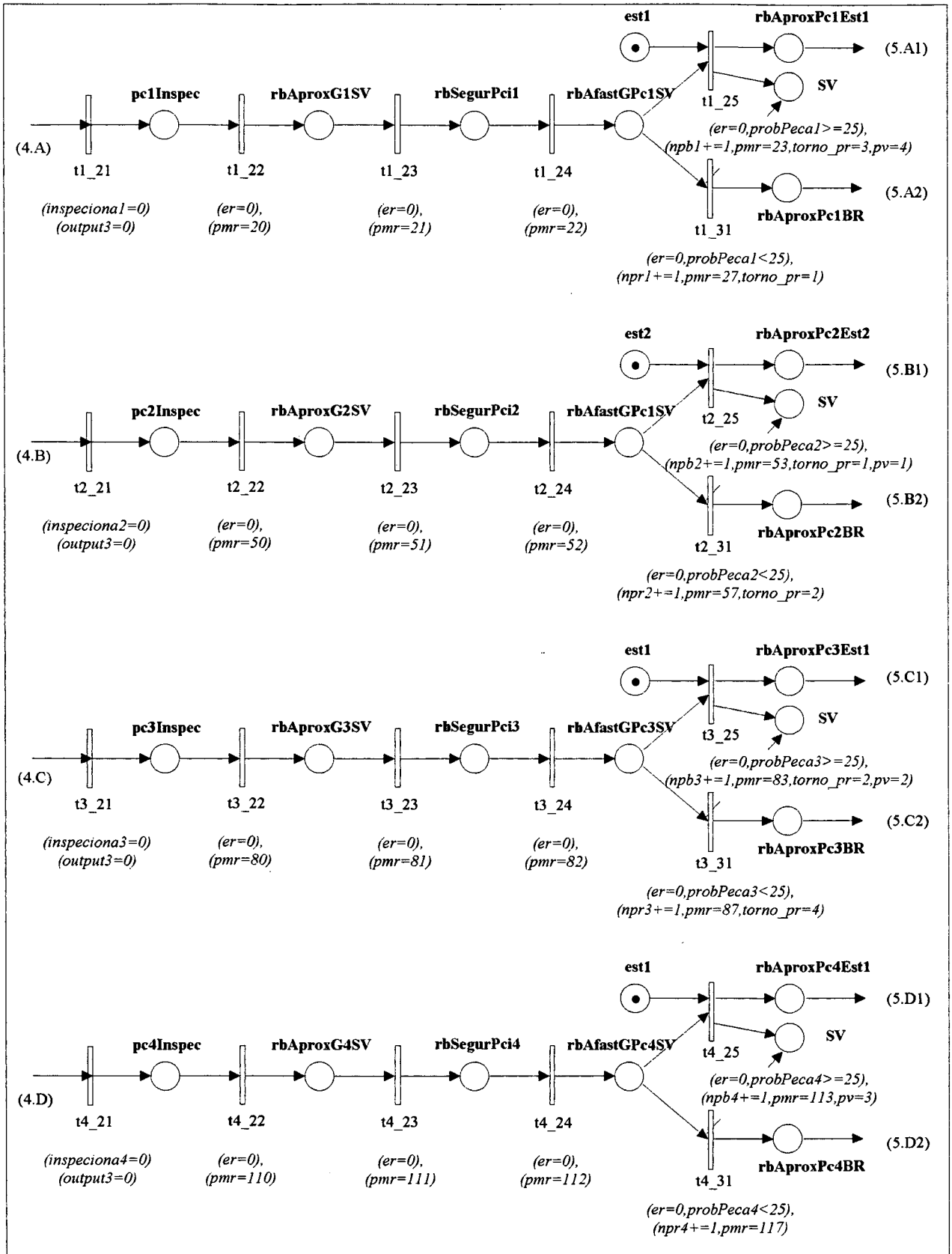


Figura 5.18 - A RPI da FMC1 - parte V

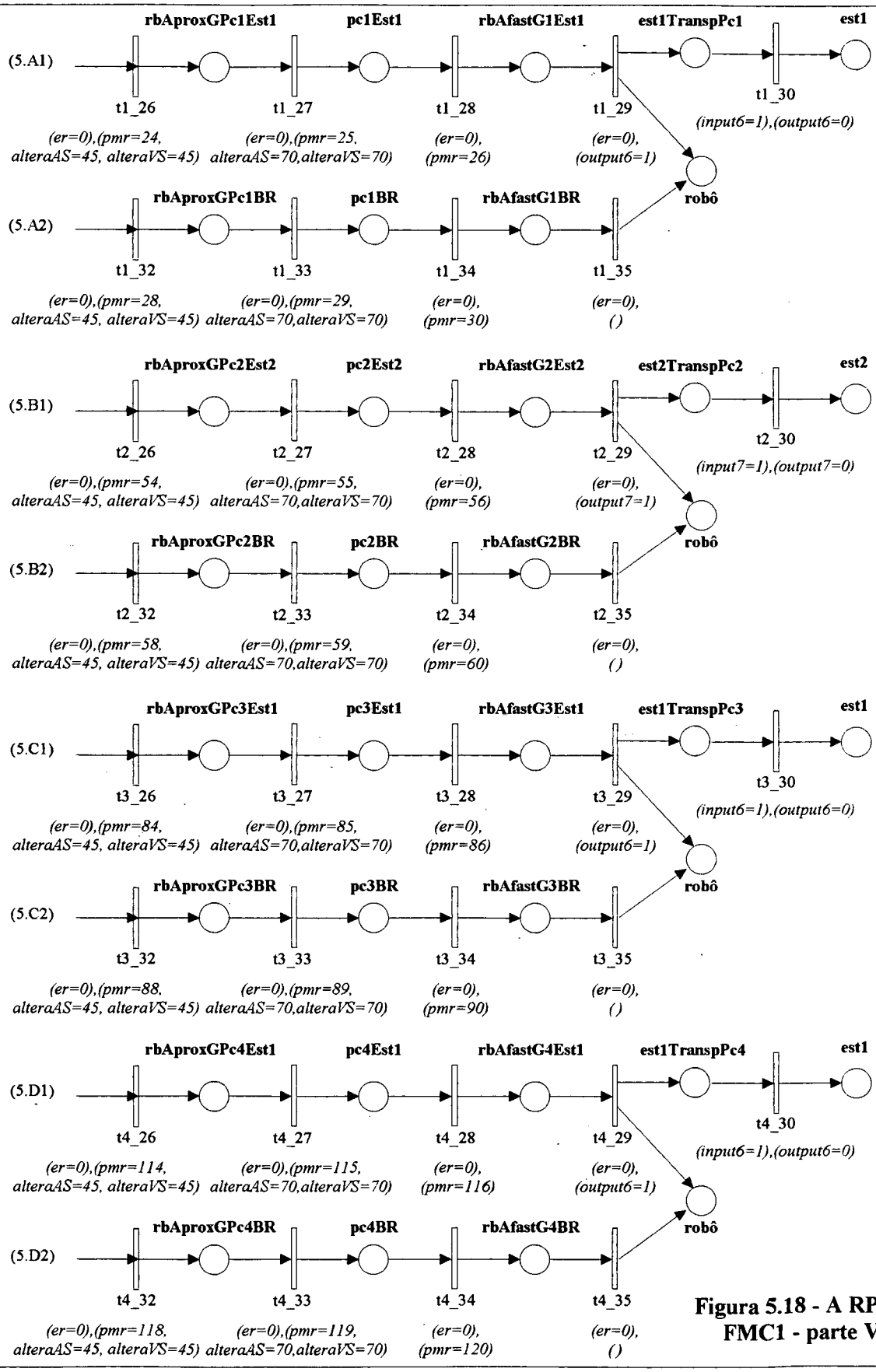


Figura 5.18 - A RPI da FMC1 - parte VI

Pode-se observar pelo desenho da rede que alguns lugares se repetem. Na realidade esses lugares são os mesmos, estão repetidos apenas porque desta maneira fica mais fácil desenhar e entender a rede. A rede também passou por várias modificações até chegar nesta forma final, inclusive nas últimas modificações algumas trasições foram retiradas, conforme pode ser observado pela não utilização dos movimentos (pmr) 12,42,72,102.

Segue a baixo uma legenda com os afixos utilizados nos nomes das transições, lugares e variáveis da rede.

* **aprox** : aproximando
* **rb** : robô
* **buf**i : *buffer* de entrada i [i=1,2,3 ou 4]
* **G** : garra do robô
* **Pc**i : peça i
* **Pcb**i : peça bruta i
* **Pci**i : peça inspecionada i
* **afast** : afastando
* **usin** : usinando
* **pciPrt** : peça i pronta
* **SV** : sistema de visão
* **segur** : segurando
* **SVInspec**i : SV inspecionando peça i
* **pciInspec** : peça i inspecionada
* **est**j : esteira j [j=1 ou 2]
* **BR** : *buffer* refugo
* **CU** : centro de usinagem
* **transp** : transportando
* **torno_pr** : torno produz
* **cu_pr** : centro de usinagem produz
* **pv** : peça da vez

Por simplificação, não foram feitas análises formais das redes (FMC1 e FMC2) desenvolvidas, para, por exemplo,

verificar as "boas propriedades" (vivacidade, limitação e reinicialização) das redes. Entretanto cada usuário, ao desenvolver o seu próprio sistema, poderá fazê-lo.

5.6 - FOTO DA CÉLULA DE FABRICAÇÃO

Tem-se uma visão geral da célula de fabricação (FMC1) através da foto mostrada na figura 5.19.

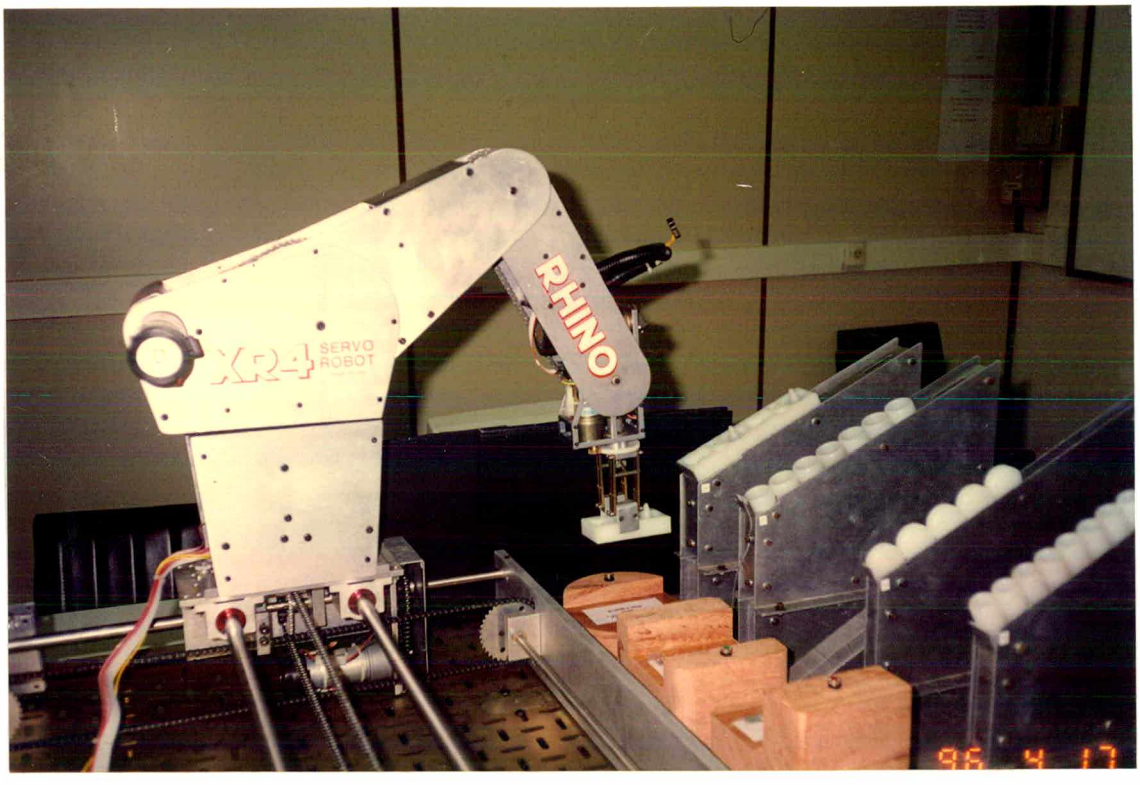


Figura 5.19 - Foto da célula flexível de fabricação

5.7 - RESUMO

Embora a FMC1 seja uma simulação de uma célula flexível de fabricação, pode-se observar que foram feitas considerações

visando aproximar o sistema produtivo proposto à realidade encontrada na indústria.

Mesmo sendo utilizada para a fabricação de apenas quatro peças, a FMC1 permite ao usuário definir os tipos de peças a serem produzidas. As limitações, no entanto, dizem repeito somente às máquinas, que serão sempre um torno, um centro de usinagem e um sistema de visão.

Pode-se observar também o papel do gerente da célula, o qual deve atender tanto as atividades de manufatura quanto ao usuário da célula.

As funções de atendimento à célula são especificadas pelo usuário através de uma rede de Petri interpretada e as funções de atendimento ao usuário são acessadas por meio de menus.

6

A Célula Flexível de Montagem

As peças consideradas BOAS após a fabricação são transportadas pelo robô para as esteiras da célula flexível de montagem (FMC2).

As peças deverão então ser transportadas pelas esteiras até uma posição próxima do robô SCARA e, quando estiverem na posição final da esteira, dentro do volume de alcance do robô, este deverá pegá-las para fazer a montagem do produto final.

Quando um produto estiver completamente montado, um outro robô (XR4) deverá retirá-lo da mesa de montagem para transportá-lo para a terceira célula flexível (FMC3). Como já dito anteriormente, a FMC3 já está operacional, e não faz parte do desenvolvimento deste trabalho.

Poderá ser observado que algumas características da FMC2 são semelhantes, ou até mesmo iguais, às da FMC1, principalmente no que diz respeito à implementação do seu programa de gerenciamento (gerente FMC2). Nestes casos, procurar-se-á ser o menos repetitivo possível, haja visto que algumas das definições a serem utilizadas já foram descritas no capítulo 5.

6.1 - Os componentes da FMC2

A célula de montagem é formada pela integração dos seguintes componentes:

- 2 esteiras transportadoras;
- 1 *pallet* transportador;

- 2 estações de montagem;
- 1 robô SCARA e o seu controlador MARK-IV;
- 1 robô XR4 e o seu controlador MARK-IV; e
- 1 microcomputador.

6.1.1 - AS ESTEIRAS TRANSPORTADORAS

As esteiras utilizadas na FMC2 são exatamente iguais. Por que então utilizar duas esteiras ao invés de apenas uma? A resposta é que, mesmo sendo iguais, elas são utilizadas de forma bem diferente, conforme será visto mais adiante.

As esteiras são alimentadas com uma tensão entre 0 e ±12 Volts e são conectadas ao controlador do robô de montagem (SCARA) nas saídas auxiliares AUX1 e AUX2.

Elas possuem servo motores DC, mas não possuem *encoders*, como os motores dos robôs e das estações de montagem. Isto faz com que não seja possível utilizar esse tipo de esteira quando a aplicação exigir um controle de posicão e/ou de velocidade.

Se os motores das esteiras possuissem tais *encoders*, não haveria necessidade de se utilizar os sensores e os posicionadores colocados no fim das esteiras.

Pode-se alterar a velocidade e o sentido de deslocamento das esteiras atribuindo-se valores entre 0 e ±100 às portas auxiliares (AUX1 e AUX2), as quais transmitem, proporcionalmente, esses valores às tensões de alimentação das esteiras. O módulo deste valor corresponderá à velocidade de deslocamento, e o seu sinal (±), ao sentido a ser adotado. Nesta implementação por exemplo, querendo-se que a esteira 1 se desloque com uma velocidade igual a 75% da velocidade máxima (VM) e no sentido da FMC2 para a FMC1, deve-se atribuir à AUX1 o valor: -75 (equivalente à 0,75VM no sentido FMC2→FMC1).

6.1.1.1 - A ESTEIRA 1

Na configuração adotada, a esteira 1 (figura 6.1) é utilizada para transportar as peças 1, 3 e 4 (figura 5.2, 5.4 e 5.5 respectivamente).

Para o transporte dessas peças, ela executa um único movimento, contínuo e numa mesma direção (FMC1→FMC2), com uma velocidade de 40% da velocidade máxima (0,40VM).

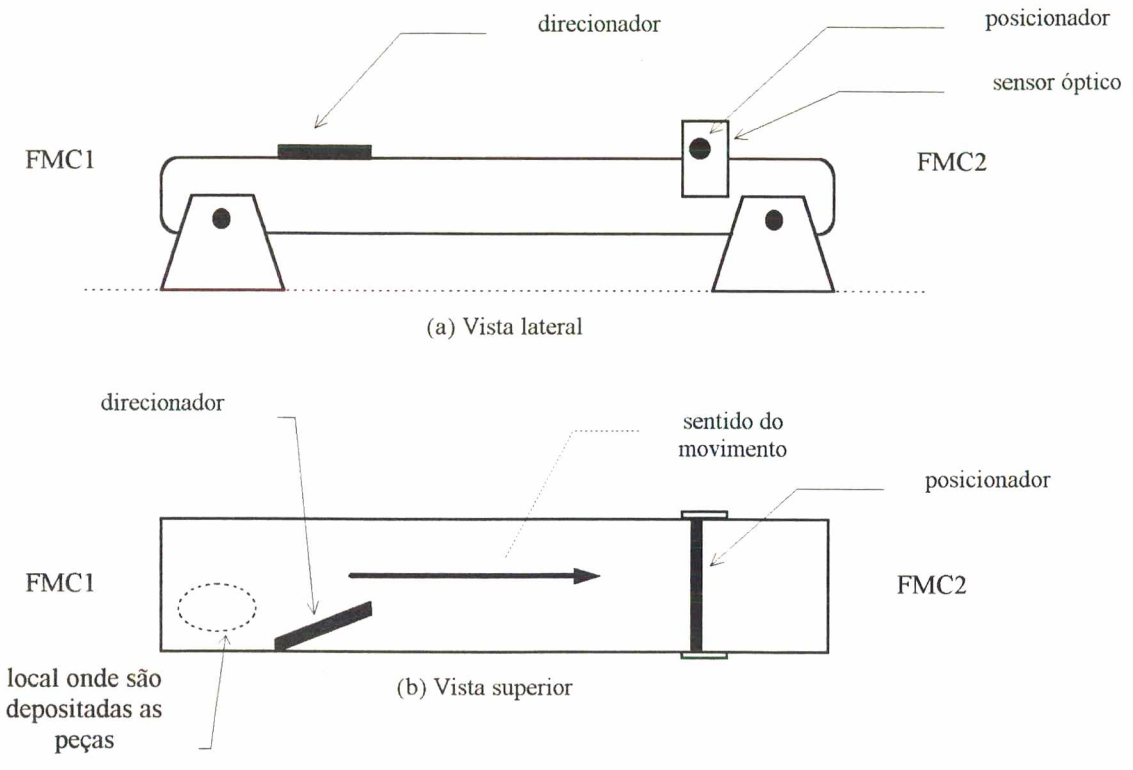


Figura 6.1 - Esteira transportadora 1

Esse valor (40%VM) foi utilizado porque, após uma série de testes, constatou-se que se a velocidade da esteira fosse maior, o direcionador derrubaria a peça do tipo 1, o que interromperia aquele ciclo de montagem. Descobriu-se que com uma velocidade menor que 50% dificilmente isto pode acontecer.

Por outro lado, as demais peças poderiam ser transportadas com uma velocidade maior, até mesmo de 100%VM, ganhando-se com isso algum tempo. Entretanto, isto não traria benefício algum à execução da célula, pois o tempo de montagem das peças é muito menor que o de fabricação e, independente da velocidade das esteiras (desde que não seja muito pequena - menor que ±20%VM), a célula de montagem ficaria ociosa, à espera de peças.

6.1.1.2 - A ESTEIRA 2 E O PALLET TRANSPORTADOR

A razão para se utilizar uma outra esteira no transporte das peças deve-se principalmente à dificuldade em se transportar a peça do tipo 2 (figura 5.3). Devido à sua geometria, quando a esteira se movia, a peça rolava, saindo da posição correta (pela mesma razão descrita em 5.2.3).

Desenvolveu-se então um *pallet* dedicado exclusivamente ao transporte dessa peça.

No *pallet* transportador, a peça fica presa entre duas pequenas hastes, que não permitem que a peça sai da sua posição inicial. Como será visto no capítulo 7, estas hastes também ajudam a corrigir os erros de posicionamento do robô, quando o mesmo tem que depositar a peça na esteira. A configuração da esteira 2, em conjunto com o pallet transportador, está ilustrada nas figuras 6.2 e 6.3.

Percebe-se que a esteira 2 deve executar um movimento do tipo "ida e volta", pois como o *pallet* não faz parte da montagem da peça, ele não deve ser retirado da esteira.

Assim, para transportar a peça para o ponto próximo do SCARA, a esteira 2 deve deslocar-se no sentido FMC1→FMC2 e, após a retirada da peça do *pallet*, deve retornar, no sentido FMC2→FMC1.

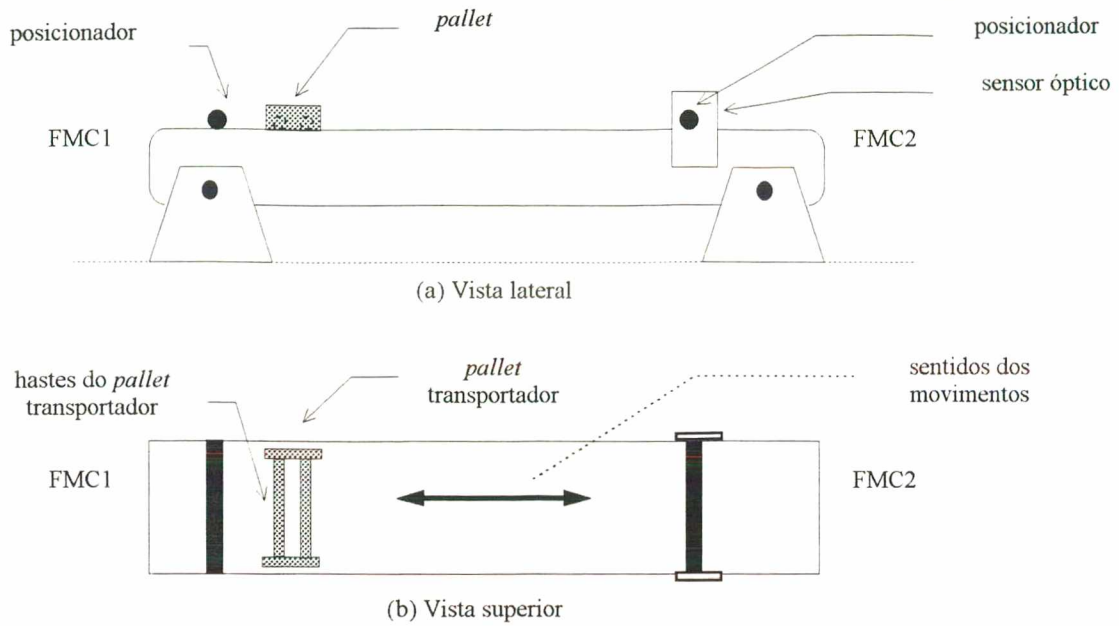


Figura 6.2 - Esteira transportadora 2

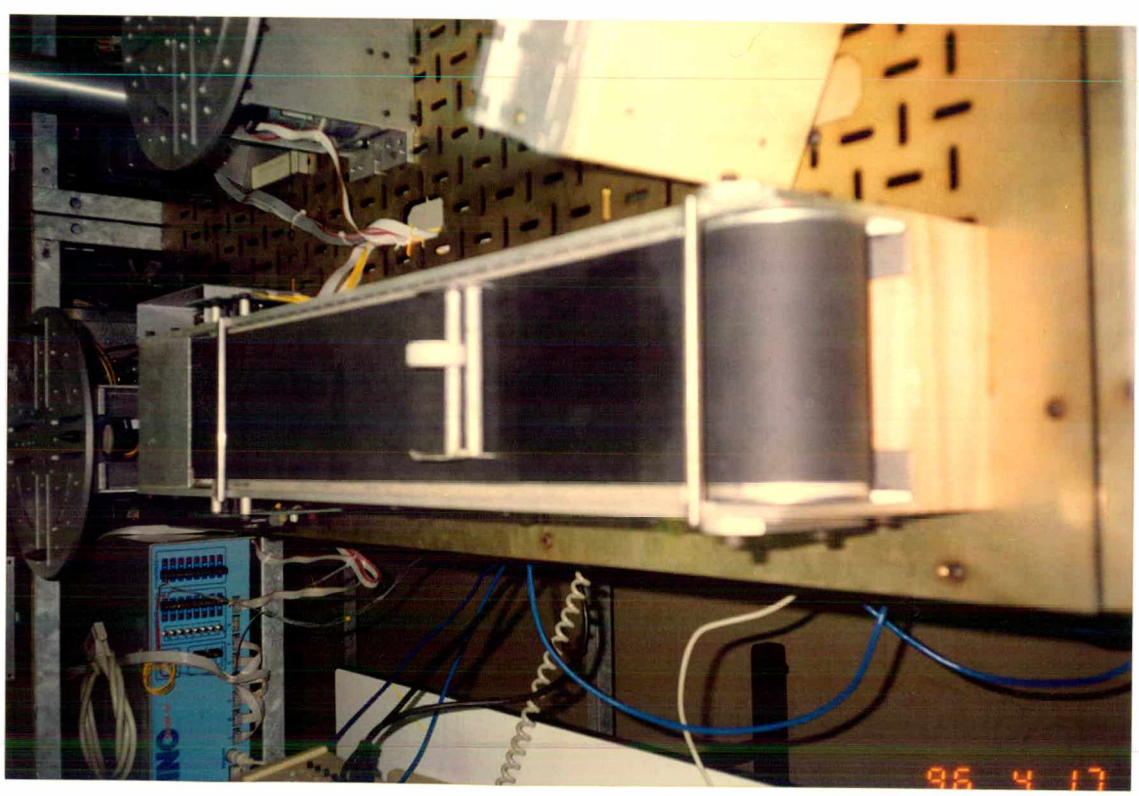


Figura 6.3 - Foto da esteira 2 transportando a peça 2

6.1.2 - AS ESTAÇÕES DE MONTAGEM

A montagem das peças na FMC2 é feita sobre uma estação de montagem (EM) (ou também chamada, mesa giratória (MG)).

Esta EM possui uma base prismática com um disco na parte superior (figura 6.4) que pode girar, tanto no sentido horário como anti-horário, comandado por um servo motor DC. Ele permite o controle de posição e velocidade pois possui no motor um *encoder* óptico.

No MARK-IV até duas EM podem a ele ser conectadas (portas 'G' e 'H'), e podem ser comandadas pelo usuário como se fossem motores do próprio robô. Porém nesta aplicação apenas uma EM será utilizada.

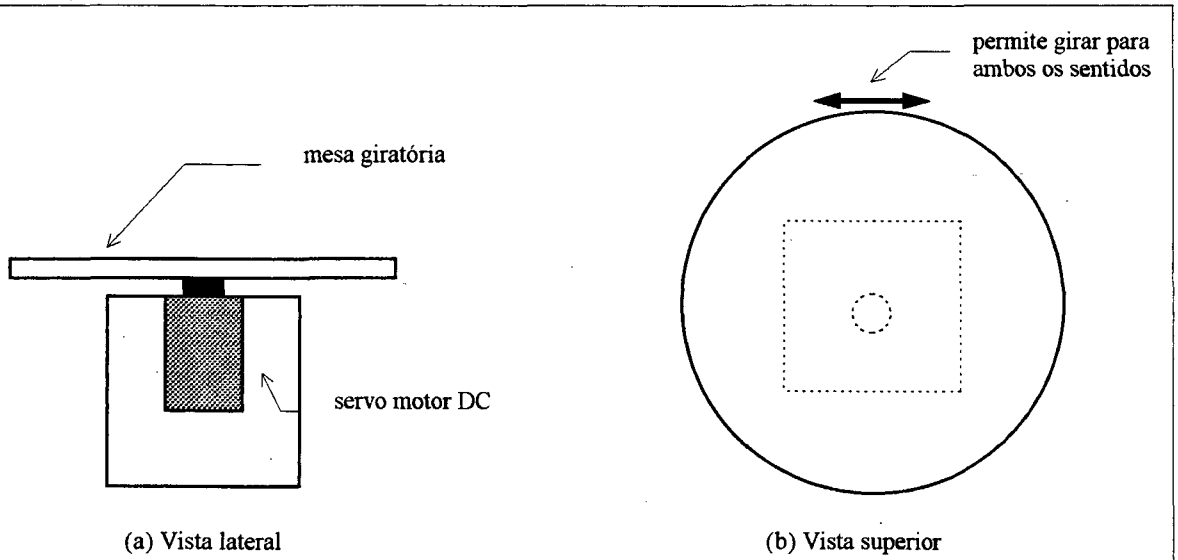


Figura 6.4 - A estação de montagem

O fato de se poder fazer o controle da posição da EM possibilita que sejam montadas várias peças simultaneamente, pois várias atividades, em diferentes estágios de montagem,

podem ser executadas à medida que a mesa gira. Isto pode melhor ser entendido com o exemplo a seguir:

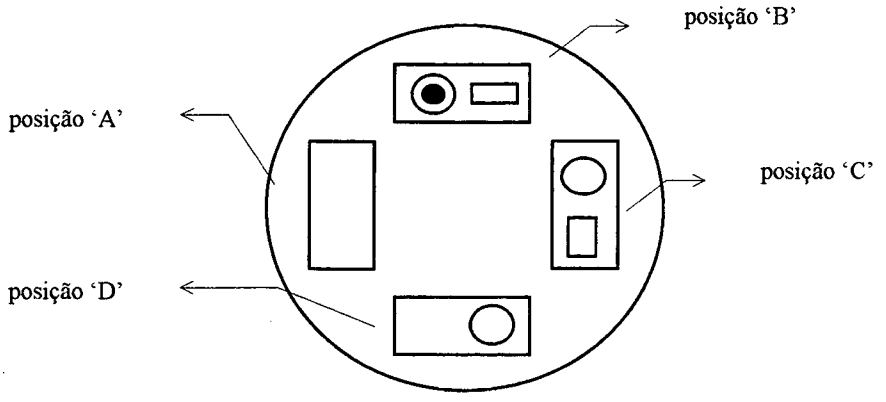


Figura 6.5 - Várias montagens simultâneas

Na estação de montagem ilustrada na figura 6.5, são executadas quatro montagens simultâneas: na posição 'A' a peça base foi colocada; na posição 'D' uma segunda peça foi montada na peça base; na posição 'C' a terceira peça já foi montada, e na posição 'B', o produto está completamente montado e já pode ser retirado da EM.

Como na FMC de fabricação existem apenas um torno CNC e deseja-se não ter estoque intermediário algum, não há possibilidade de haver mais de uma peça sendo montada ao mesmo tempo. Mas numa outra aplicação, o usuário pode definir, por exemplo, que os quatro tipos de peças devem ser produzidos em lotes de quatro, isto é, 4 peças do tipo 1, quatro do tipo 2, etc, dessa forma serão montadas quatro peças simultaneamente. Uma outra situação seria se existissem mais máquinas na FMC. Neste caso várias montagens, inclusive utilizando as duas EMs, poderiam ser executadas.

6.1.3 - O COMPUTADOR CENTRAL DA FMC2

O computador central da célula de montagem possui as mesmas funções que o CCC de fabricação, com a diferença que deve integrar tipos diferentes de dispositivos: dois robôs (XR4 e SCARA), duas esteiras e duas estações de montagem.

O CCC utilizado na FMC2 foi do tipo 486, com 50 MHz de velocidade, disco rígido de 240 MBytes e memória de 4 MBytes.

6.2 - INTEGRAÇÃO DOS COMPONENTES

Neste ponto serão tratados algumas questões relativas à integração dos componentes da FMC2. Não serão considerados aspectos referentes à parte de programação e utilização, apenas a parte de *hardware* será abordada. Alguns problemas encontrados, junto com as soluções adotadas também serão descritos.

Como na FMC1, os dispositivos da célula são integrados fisicamente através do controlador do robô. Na FMC2, todos os componentes são conectados ao MARK-IV do robô de montagem (SCARA), não sendo preciso utilizar o outro controlador (do XR4).

O MARK-IV do SCARA é ligado ao CCC através da porta serial (RS-232C) COM1 e o controlador do XR4, através da serial COM2.

6.2.1 - AS CONEXÕES NOS CONTROLADORES DOS ROBÔS

As configurações dos controladores dos robôs SCARA e XR4 estão mostradas nas figuras 6.6 e 6.7 respectivamente.

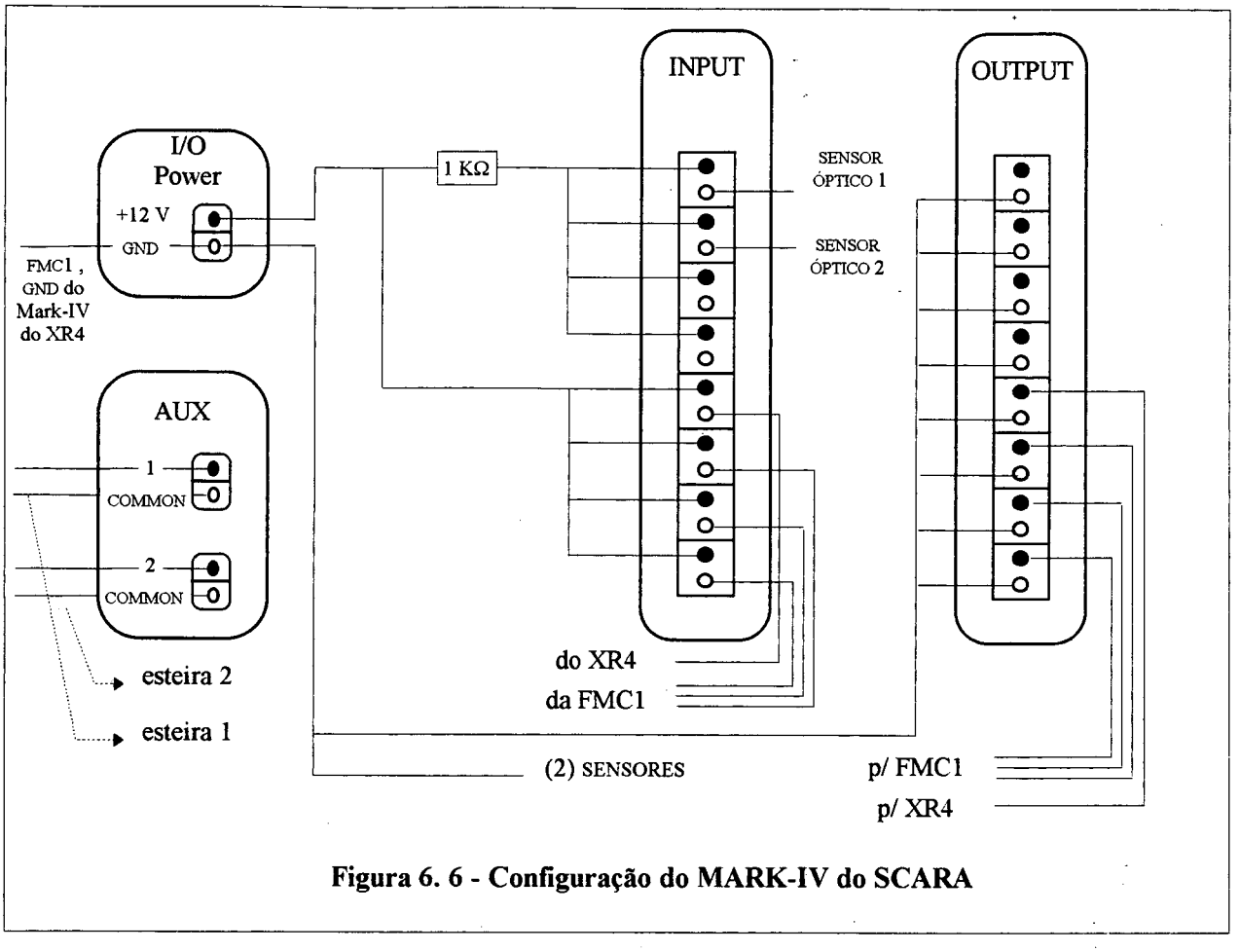


Figura 6. 6 - Configuração do MARK-IV do SCARA

Deve-se mencionar aqui que as conexões relativas à integração dos sensores ópticos das esteiras não precisaram ser feitas pois estas já haviam sido implementadas anteriormente.

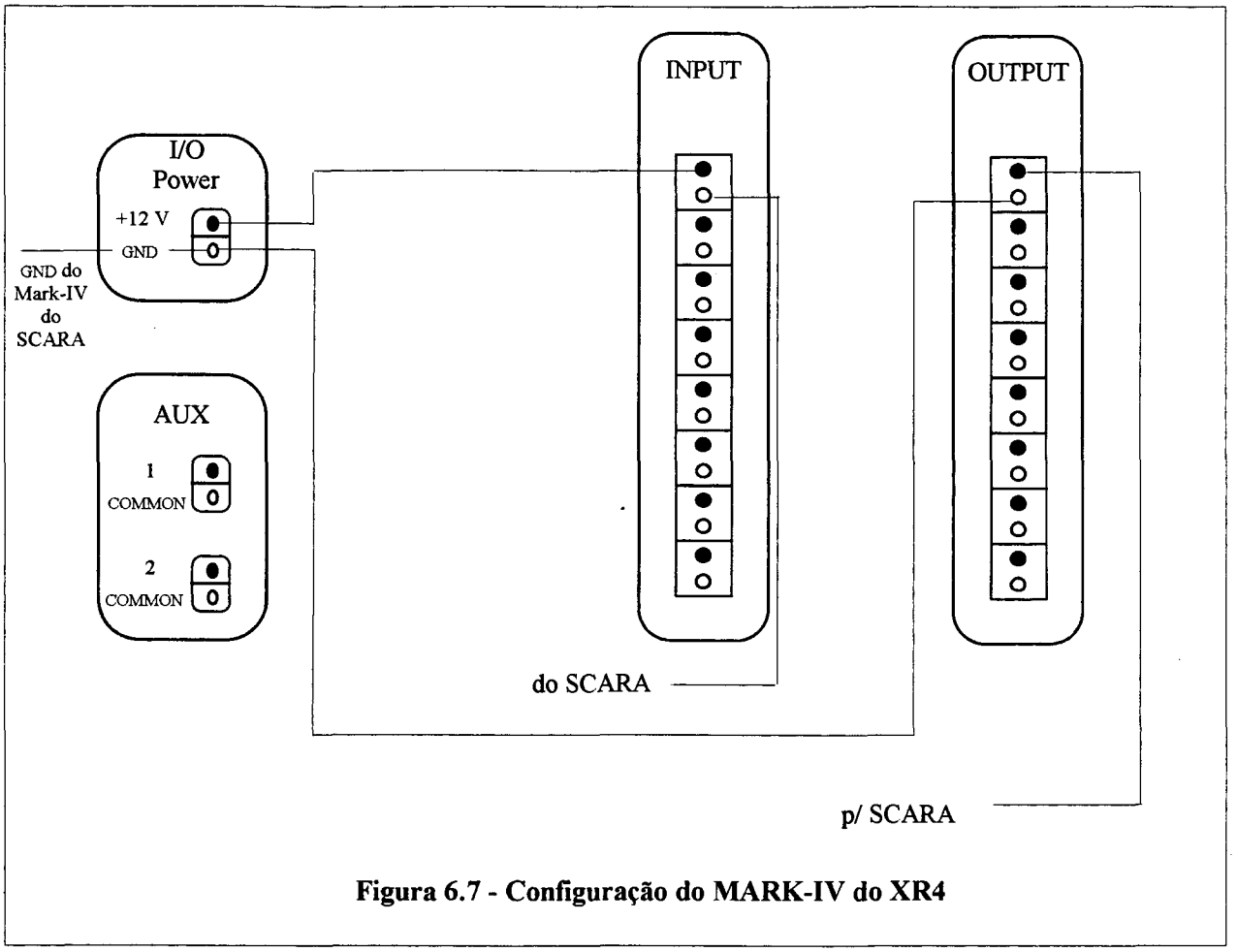


Figura 6.7 - Configuração do MARK-IV do XR4

Como dito anteriormente, todos os dispositivos da célula puderam ser integrados no controlador do SCARA, e portanto, não foi preciso utilizar as entradas e saídas do MARK-IV do XR4. Entretanto, fez-se uma pequena integração entre o controlador do XR4 e o controlador do SCARA, para o caso de um possível usuário poder vir a precisar destes sinais.

6.2.2 - Problemas e soluções

Na integração da célula de montagem ocorreram menos problemas do que na célula de fabricação, principalmente porque o SCARA, que é o robô que trabalha quase todo o tempo (o outro robô é utilizado apenas para transportar o produto final para a

FMC3), possui uma precisão e repetibilidade bem melhor que o robô da FMC1.

A seguir faz-se uma rápida descrição de alguns desses problemas junto com suas respectivas soluções.

(a) Perda de informações nos sinais de entrada (INPUTS) no controlador do SCARA.

Descobriu-se que havia a necessidade de se interligar os sinais 'terra' (GND) dos três controladores.

(b) A garra do SCARA também não possuia abertura suficiente para pegar as peças 3 e 4. Desenvolveu-se então um ampliador de garra para ele, além de um outro também para o XR4.

(c) Algumas vezes a montagem da peça 2 na peça prismática não era feita corretamente devido à sua geometria, à precisão de montagem exigida e aos erros do SCARA.

Depois de vários testes constatou-se que a melhor solução era não posicionar a peça 2 na peça prismática, mas sim soltá-la, de uma pequena altura e num ângulo tal que, quando ela tocasse a peça, automaticamente se ajustasse, encaixando-se corretamente[1] na peça base(figura 6.8).

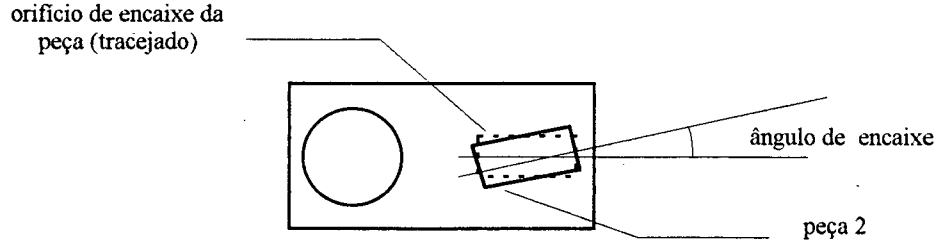


Figura 6.8 - Ângulo de encaixe da peça 2

[1] Isto só acontece porque tanto a peça quanto o orifício onde a mesma deve ser montada possuem os cantos chanfrados.

(d) Os sensores ópticos colocados no fim das esteiras são utilizados para informar ao gerente da célula que as peças já estão no fim das esteiras, para que ele possa saber quando interromper os seus movimentos. Após a parada das esteiras, as peças deveriam estar sempre numa mesma posição, mas, em tempo de execução, constatou-se que quase sempre elas paravam em pontos bem diferentes.

Desenvolveu-se então um dispositivo chamado de *posicionador*. O posicionador "segura" as peças, impedindo-as de sair da posição desejada. Ele, como o direcionador (capítulo 7), tem a forma de uma pequena haste em alumínio, conforme pode ser observado na figura 6.3.

Esses posicionadores são muito importantes na FMC2 porque o robô que faz a montagem não pode, de forma alguma, receber as peças em posições variadas. Uma certa flexibilidade de precisão é até admissível (na realidade sempre haverá imprecisão), mas desde que seu valor esteja dentro dos limites de construção do robô.

6.3 - O GERENTE FMC2

O leitor provavelmente achará muitas semelhanças entre esta seção e a seção 5.4, pois o *software* de gerenciamento da célula flexível de montagem (gerente FMC2) possui características e atribuições muito parecidas com as do gerente da célula de fabricação.

As maiores diferenças entre eles estão nas funções de atendimento à célula (FAC) e nas funções de atendimento ao usuário (FAU). A estrura de execução interna e as variáveis externas são da mesma forma bastante semelhantes.

6.3.1 - As FAC do gerente FMC2

Também na célula de montagem uma rede de Petri interpretada é responsável pelo modelamento das atividades relacionadas à montagem das peças. Essa RPI segue as mesmas regras de utilização e construção da FMC1.

Ela deve ser escrita sob a forma de uma arquivo texto chamado de "FMC2.RDP" e deve obedecer a sintaxe definida pelo *software* de rede de Petri mencionado no capítulo 3. No anexo III, tem-se a descrição dos arquivos com as RPIs da FMC1 e da FMC2, que poderão ser utilizados como referência ao desenvolvimento de outras aplicações.

As principais FAC do gerente FMC2 são:

1. Comandar e controlar os robôs, as esteiras e as estações de montagem.
2. Coordenar a sequência das atividades a serem executadas na célula.
3. Estabelecer a integração e o sincronismo com a célula de fabricação.
4. Permitir a alteração da configuração da célula, quando, por exemplo, novos tipos de peças tiverem que ser produzidos. Essas novas peças poderão ser completamente diferentes das peças montadas nesta aplicação, inclusive com diferentes números de componentes.

6.3.2 - As FAU do gerente FMC2

A estrutura de navegação através dos menus no gerente FMC2 tem que ser maior que a do gerente FMC1, porque, mesmo sem os três dispositivos_máquinas, existem dois controladores MARK-IV, enquanto que na FMC1 existe apenas um.

As funções de atendimento ao usuário já estão detalhadas no capítulo anterior, e não serão portanto descritas

novamente aqui. Basta apenas que o leitor observe com atenção a nova estrutura, e observe também os respectivos nomes das opções nos menus (figura 6.9).

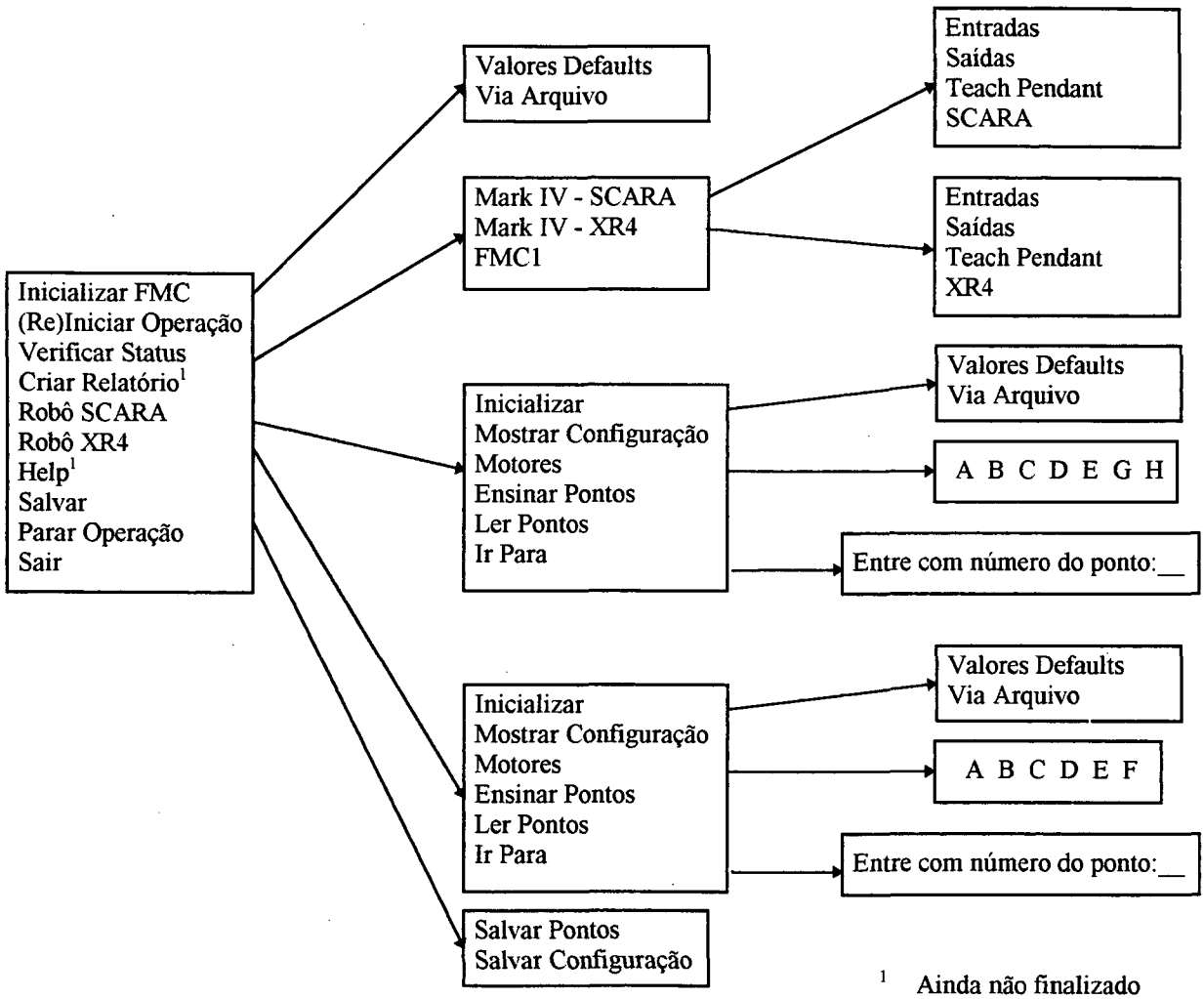


Figura 6.9 - Estrutura de navegação do gerente FMC2

6.3.3 - A ESTRUTURA DE EXECUÇÃO DO GERENTE FMC2

A figura 6.10 mostra como ficou desenvolvida a estrutura de execução interna do gerente FMC2. Comparando-a com a estrutura do gerente FMC1 percebe-se que há pouca diferença

entre eles, pois essas diferenças estão contidas pincipalmente nas FAU e FAC.

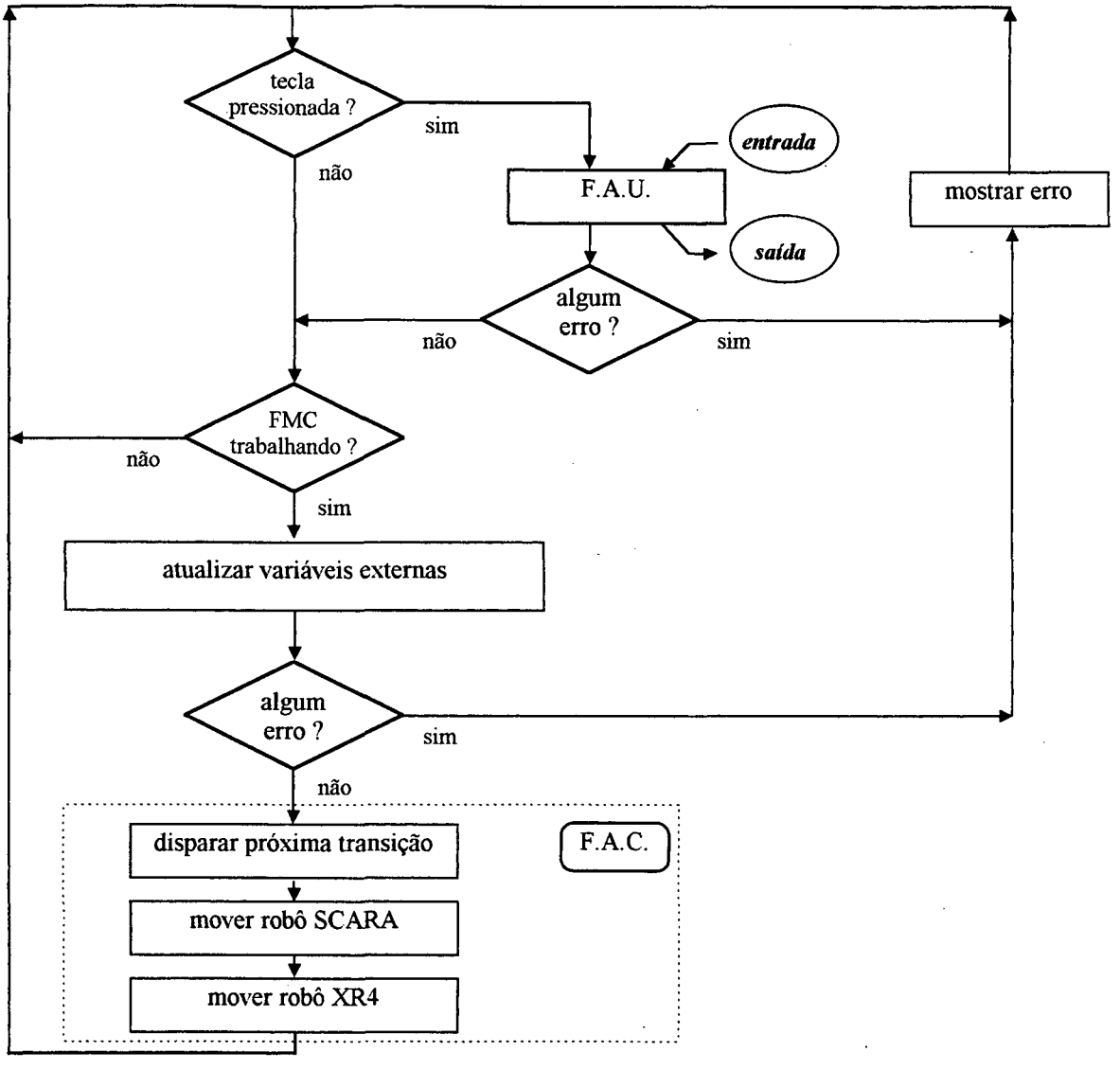


Figura 6.10 - Estrutura interna do gerente FMC2

6.3.4 - AS VARIÁVEIS EXTERNAS

O conjunto das variáveis externas utilizadas pelo gerente FMC2 é praticamente o mesmo do gerente FMC1, com a diferença que, por haverem dois robôs, o número de VEs é o dobro.

Através das VEs o usuário comanda os movimentos do SCARA e do XR4, das esteiras e das mesas giratórias, faz a leitura das entradas e ativa as saidas dos controladores.

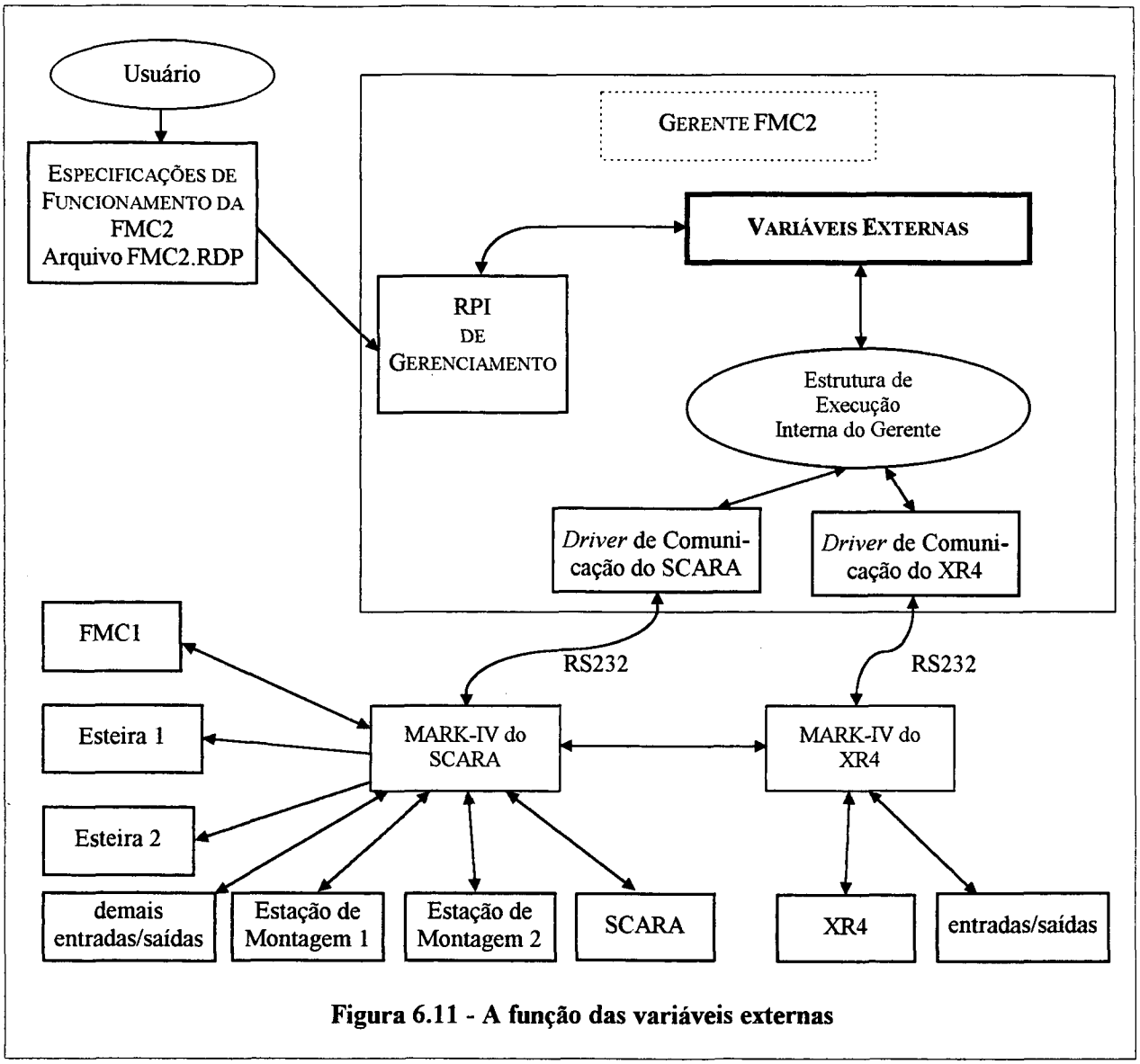


Figura 6.11 - A função das variáveis externas

As variáveis externas da FMC2 têm exatamente as mesmas funções que as VEs da FMC1, e portanto, não há necessidade de se repetir as explicações feitas na seção 5.4.4.

Entretanto, o nome das variáveis tiveram que ser um modificados pelo fato de existirem dois controladores. Definiu-se então que uma variável com o afixo 's' estaria se referindo ao controlador do SCARA, e uma com um 'x', ao controlador do XR4.

Ássim, por exemplo, a variável "input1s" refere-se à entrada 1 do MARK-IV do SCARA e "input1x" à entrada 1 do XR4. O anexo IV traz uma listagem completa das variáveis externas utilizadas na FMC1 e na FMC2.

6.4 - A UTILIZAÇÃO DA FMC2

Dentro da configuração do sistema produtivo proposto a FMC2 pode montar quaisquer tipos de peças, desde que estejam dentro da capacidade do robô SCARA. Nesta aplicação entretanto, faz-se a montagem das quatro peças produzidas pela célula de fabricação.

Essa montagem obedecerá uma ordem constante, pré-definida, que é: montar a peça 4 (peça base), depois a peça 3, então a peça 2 e, por último, a peça 1. Esta sequência poderia ser diferente, mas, em nenhuma outra situação, o tempo de montagem seria menor. Na melhor hipóteses este tempo poderia ser igual ao tempo atual. Imagine, por exemplo, se ao invés de chegar primeiro a peça base, chegasse a peça 1 (ou a 2 ou 3). Essa peça seria depositada numa posição qualquer na EM, aguardando a chegada da peça base, para só então poder ser montada.

A sequência das operações executadas pela FMC2 são:

1°. Transporte da peça prismática (peça base) pela esteira 1.

2°. Após a chegada da peça prismática no ponto final da esteira, o robô SCARA pega a peça e a deposita na estação de montagem.

3°. Transporte da peça 3 pela esteira 1.

4°. Após a chegada da peça 3 no ponto final da esteira 1, o SCARA pega esta peça e a deposita na peça base.

5°. Transporte da peça 2 pela esteira 2 (peça sobre o *pallet* transportador).

6°. Após a chegada da peça 2 no ponto final da esteira 2, o SCARA a deposita na peça base.

7°. Transporte da peça 1 pela esteira 1.

8°. Depois da chegada da peça 1 no ponto final da esteira, o robô pega esta peça e a deposita sobre a peça 3, que está na peça base.

9°. Por último o robô XR4 pega o produto completamente montado (produto final) e o transporta à terceira célula.

Embora a execução das etapas aconteça sequencialmente, pode acontecer em algum momento que as duas esteiras estejam transportando peças ao mesmo tempo.

6.4.1 - A RPI DESENVOLVIDA

Para a configuração do sistema proposto, a rede de Petri interpretada desenvolvida está representada na figura 6.11. Se o leitor desejar, pode utilizar o arquivo da rede (FMC2.RDP) no anexo III como fonte exemplo para consulta durante a implementação da sua própria célula de montagem.

Também nesta representação, alguns lugares estão repetidos, para facilitar a compreensão e o desenho da rede.

Nas condições e ações das transições T2_07 e T2_08, estão representados o retorno da esteira 2, pois conforme já descrito, ela transporta um *pallet* que, após levar uma peça 2, deve retornar à posição inicial, para poder transportar a peça seguinte. Como não se utilizou nenhum sensor para parar a esteira 2 quando esta atingisse o ponto inicial, foi implementado um contador para dar o sinal de parada. Assim, quando a variável '*conta*' for igual a 1, o gerente começa a incrementar a variável

'contador', que, ao atingir um tempo de 17 segundos (o tempo de retorno da esteira 2 com 40%VM é de aproximadamente 15 segundos) habilita T2_08, que, ao disparar, interrompe o movimento da esteira (*aux2s*=0).

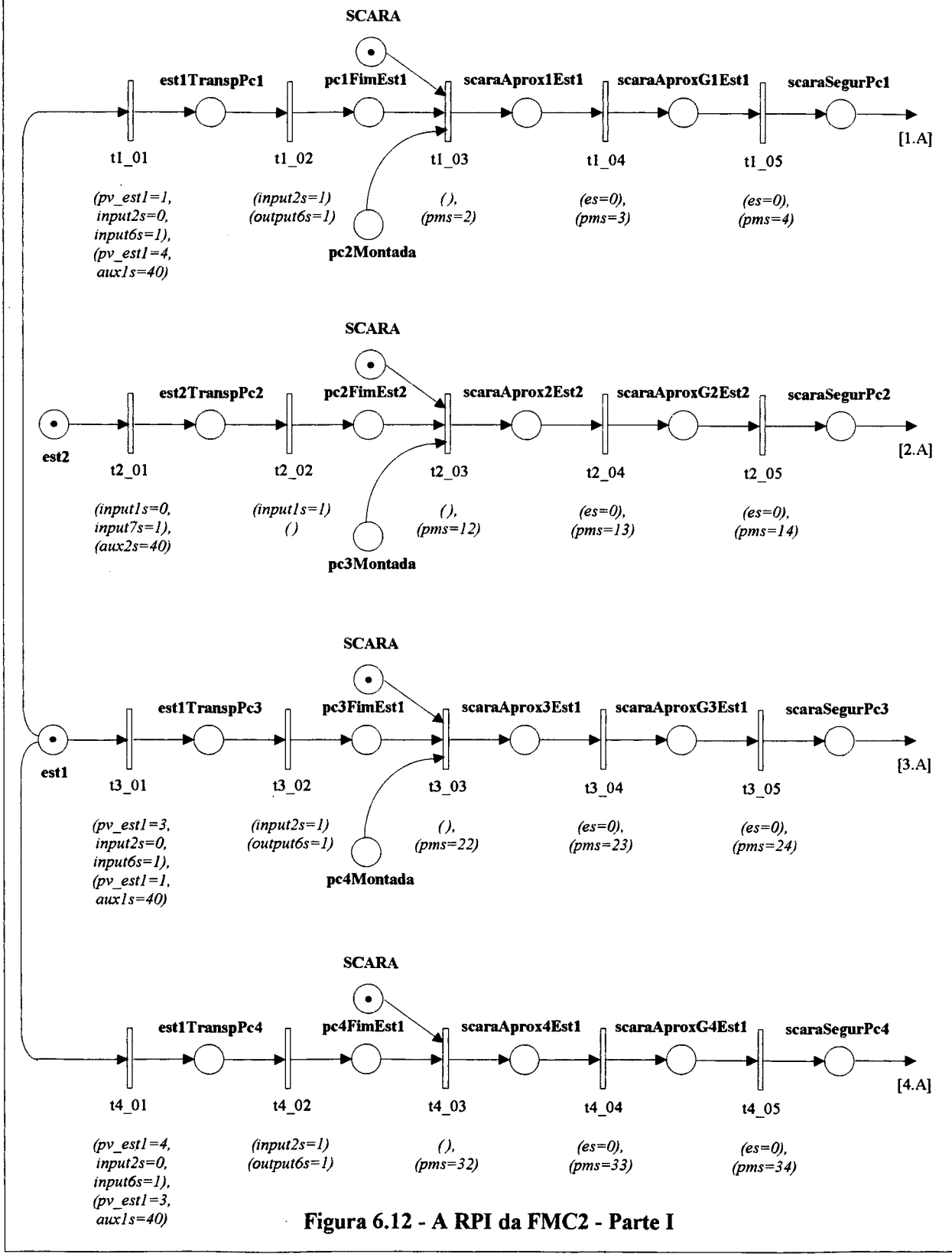


Figura 6.12 - A RPI da FMC2 - Parte I

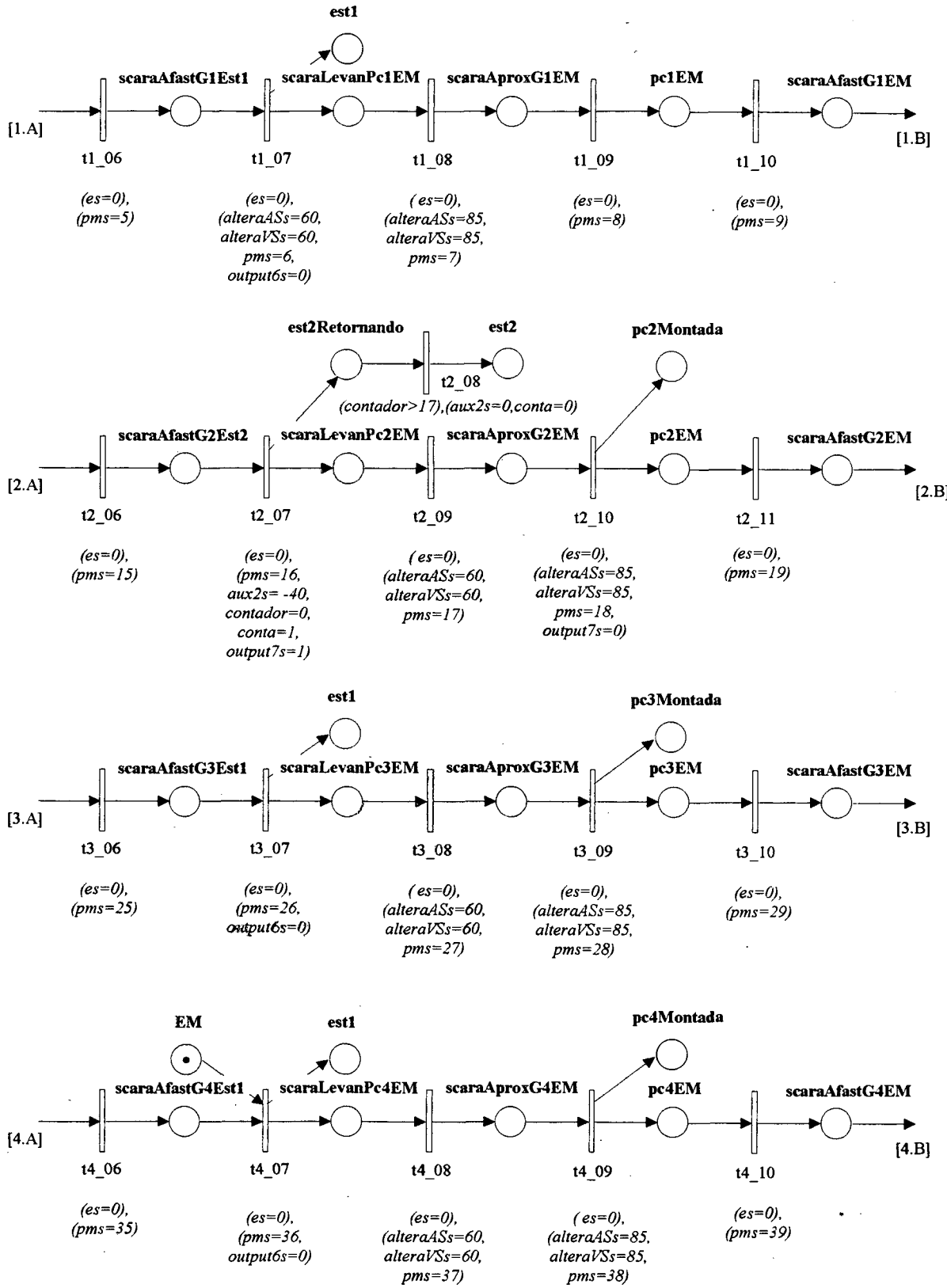


Figura 6.12 - A RPI da FMC2 - Parte II

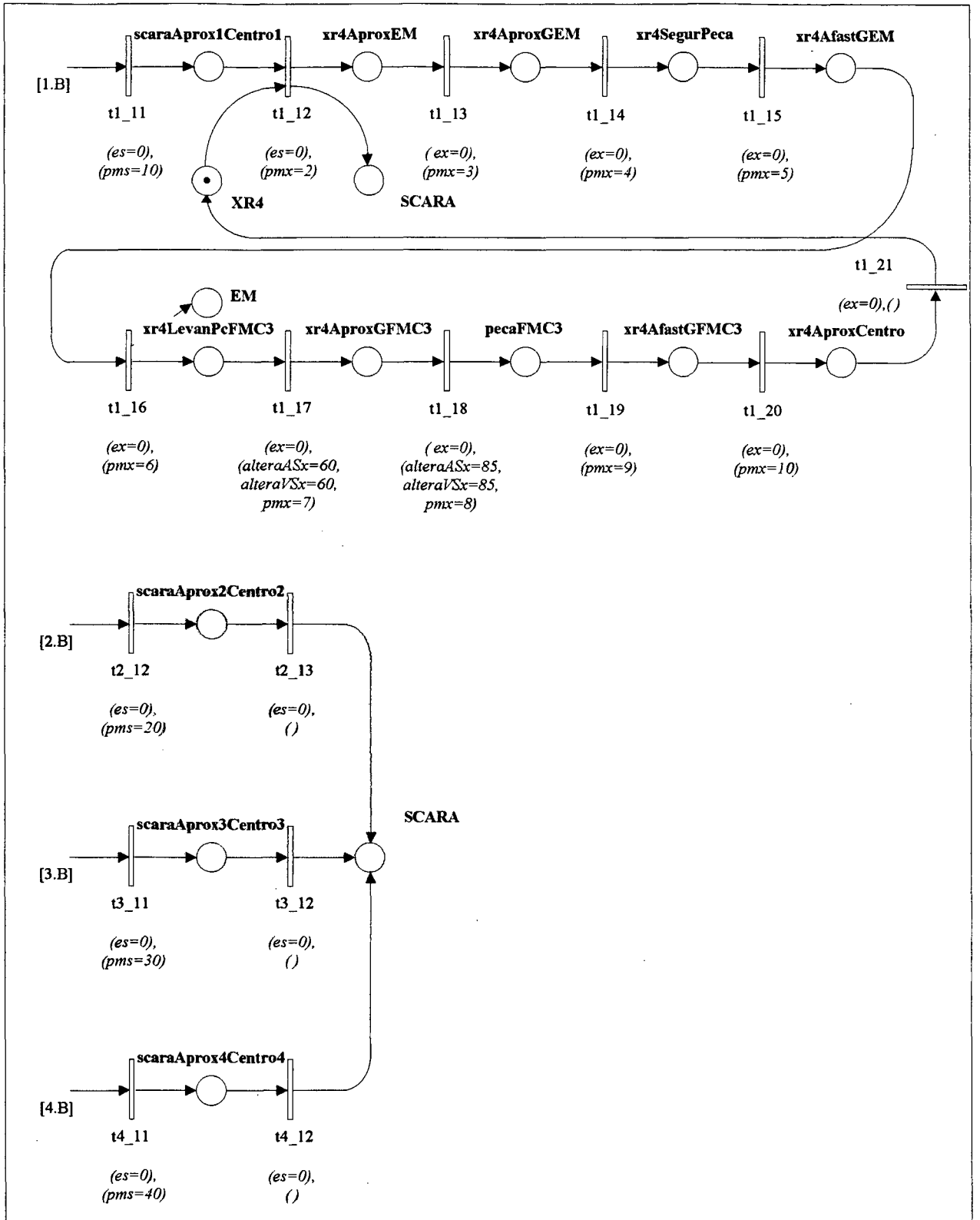


Figura 6.12 - A RPI da FMC2 - Parte III

6.5 - FOTOS DA FMC2

Nas figuras 6.13, 6.14 e 6.15 são mostradas algumas fotos da célula de montagem.

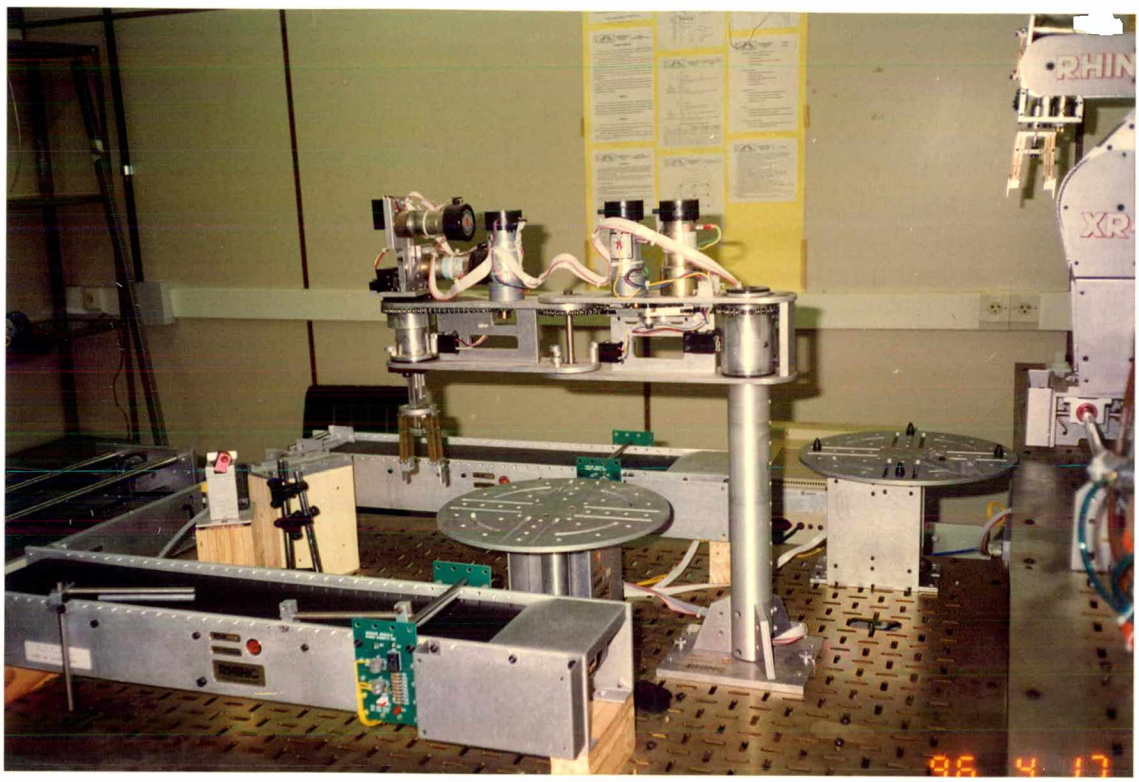


Figura 6.13 - Foto da célula flexível de montagem

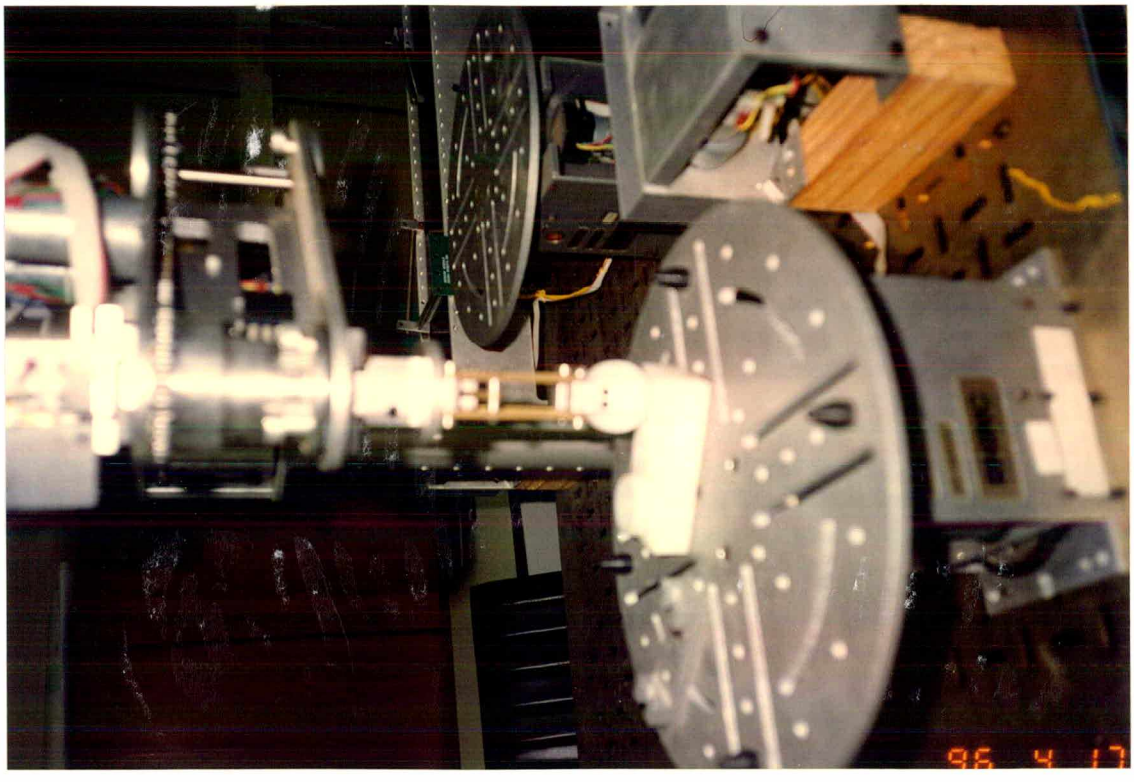


Figura 6.14 - Foto: Montando peça 2

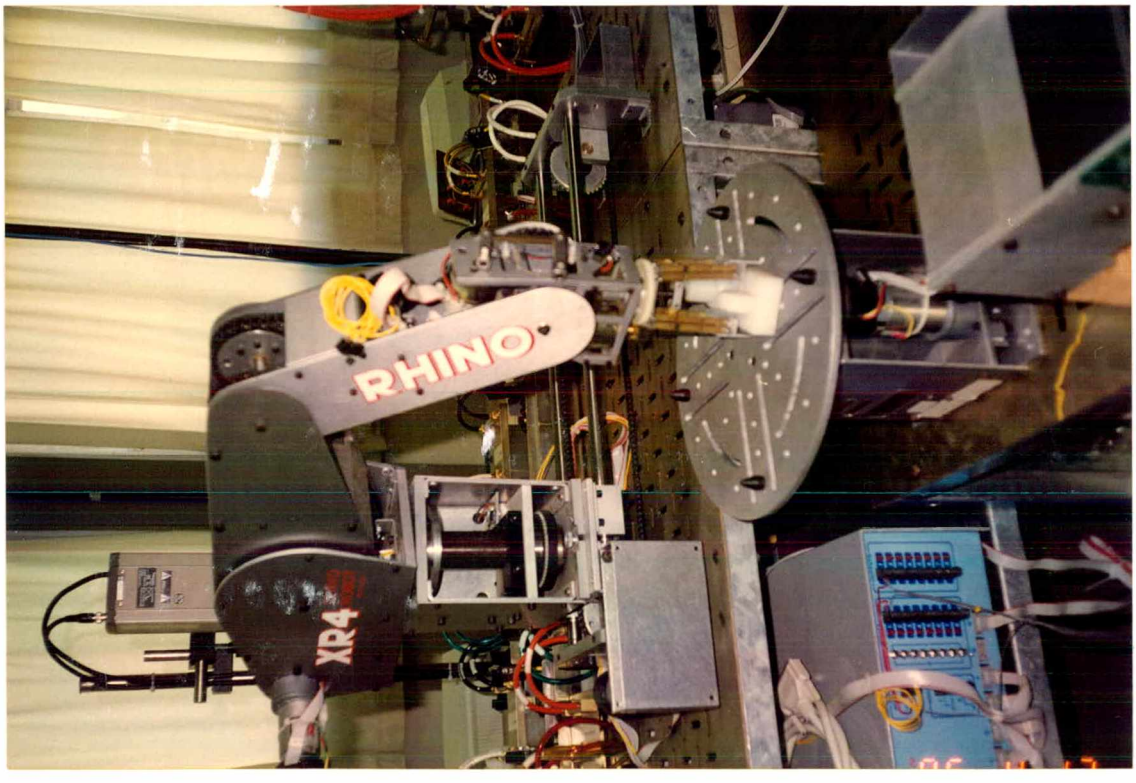


Figura 6.15 - Foto: XR4 pegando produto final para levá-lo à FMC3

6.6 - RESUMO

Apesar da FMC1 ser uma simulação de uma célula flexível de usinagem, a FMC2 por sua vez faz de fato a montagem de peças. Nesta, os problemas causados pela imprecisão do robô da célula de fabricação têm que ser solucionados, pois para que a montagem seja feita corretamente, os erros oriundos da FMC1 não poderão existir.

A FMC2 foi formada pela integração de dois robôs (SCARA e um XR4), duas esteiras e duas estações de montagem e, nesta aplicação, faz-se a montagem das quatro peças produzidas pela célula de fabricação.

Também no gerente FMC2 as funções de atendimento à célula são modeladas por redes de Petri interpretadas e as funções de atendimento ao usuário acessadas por meio de opções nos menus.

7

INTEGRAÇÃO DAS FMCs

No capítulo 2 foi mencionado que o gerente FMS é quem faz a integração lógica e física das FMCs num sistema flexível de manufatura.

Mesmo não possuindo um gerente FMS e nem podendo ser considerado como um sistema flexível, é preciso, de alguma forma, fazer a integração das células de fabricação e de montagem, pois:

(a) É preciso estabelecer um sincronismo entre a execução das tarefas nas FMCs. Por exemplo, as esteiras devem estar disponíveis quando o robô da FMC1 for depositar as peças na FMC2.

(b) A saída da FMC de fabricação tem que ser aceitável como entrada para a célula de montagem; isto é, a FMC1 deve produzir peças que possam ser montadas pela FMC2 e, além disso, deve produzir numa velocidade tal que a FMC2 possa acompanhar. Vale lembrar que nesta aplicação não é considerada a possibilidade de haver estoques intermediários.

(c) Deve ser definido a priori a ordem de saída das peças da FMC1 e a ordem de entrada das mesmas na FMC2. Para a perfeita integração, esta ordem deve ser sempre constante, ou então o gerente FMC1 deverá informar ao gerente FMC2 que tipo de peça ele está enviando a cada momento.

A integração física das células é feita através de um cabo formado por três pares de fios, ligando-se três entradas do controlador do XR4 da fabricação (portas 6,7 e 8) às três saídas do controlador do SCARA (portas 6,7 e 8), e vice-versa (figura 7.1).

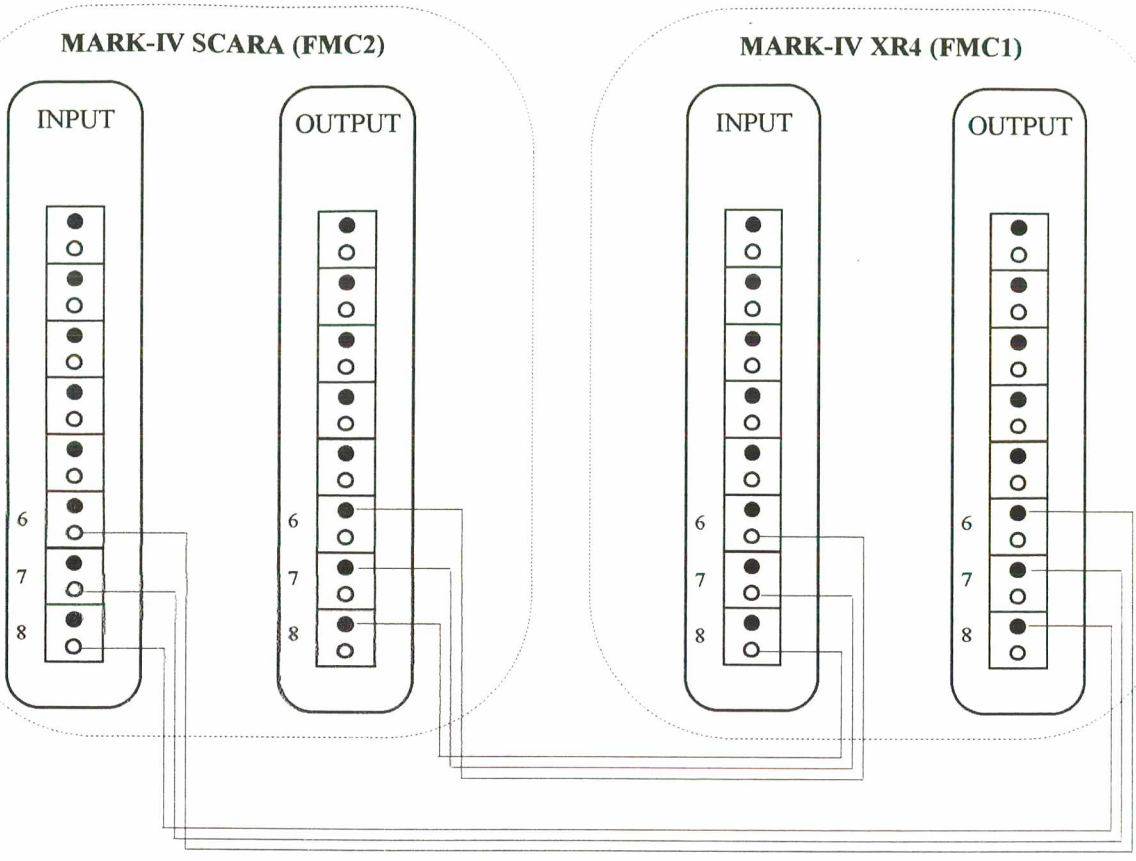


Figura7.1 - Integração física FMC1-FMC2

A nível lógico, a comunicação entre o gerente FMC1 e o gerente FMC2 é feita através da troca de sinais binários (ON/1:passando corrente e OFF/0:sem corrente).

Através desses sinais, um gerente pode receber (ou enviar) até oito tipos de informações, formadas a partir da combinação mostrada na figura 7.2.

portas:	6	7	8	
[	OFF	OFF	OFF	]⇒ informação 1
[	OFF	OFF	ON	]⇒ informação 2
[	OFF	ON	OFF	]⇒ informação 3
[	OFF	ON	ON	]⇒ informação 4
[	ON	OFF	OFF	]⇒ informação 5
[	ON	OFF	ON	]⇒ informação 6
[	ON	ON	OFF	]⇒ informação 7
[	ON	ON	ON	]⇒ informação 8

Figura 7.2 -Estrutura de informações

Esses seis sinais (três FMC1 para FMC2 e três FMC2 para FMC1) foram implementados para satisfazer a qualquer configuração do sistema. Entretanto, nem todos precisam ser necessariamente utilizados, como é o caso desta aplicação, por exemplo, onde são utilizados apenas quatro dessas seis conexões.

7.1 - O PROTOCOLO DE COMUNICAÇÃO UTILIZADO

A maneira como foi definida a comunicação entre os gerentes FMC é tão simples que talvez seja um pouco de ousadia chamá-la de "um protocolo de comunicação". Mas, visando sempre aproximar o sistema proposto de um sistema encontrado nas indústrias, decidiu-se chamá-lo assim.

Neste trabalho, o usuário define, através da interpretação nas RPIs (condições e ações), como deve ser estabelecida a comunicação entre os gerentes.

As transições da FMC1 e da FMC2 mostradas nas figuras 7.3 e 7.4, respectivamente, mostram como foi definido o protocolo utilizado pelos gerentes.

Deve-se perceber que o objetivo central da integração das células desta aplicação é verificar quando as esteiras da FMC2 estarão disponíveis à FMC1, pois o robô não poderá

depositar as peças nas esteiras se elas estiverem se movendo ou, no caso da esteira 2, se o *pallet* ainda não estiver na posição correta para poder receber a peça.

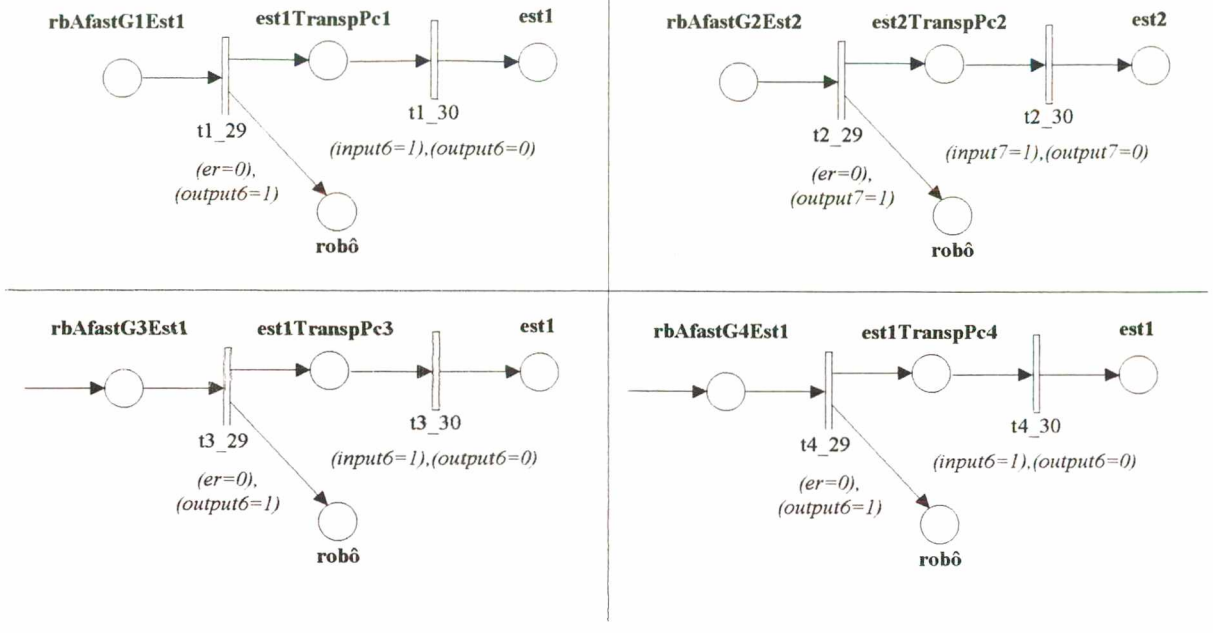


Figura 7.3 - Transições na RPI da FMC1 que fazem a comunicação com o gerente FMC2

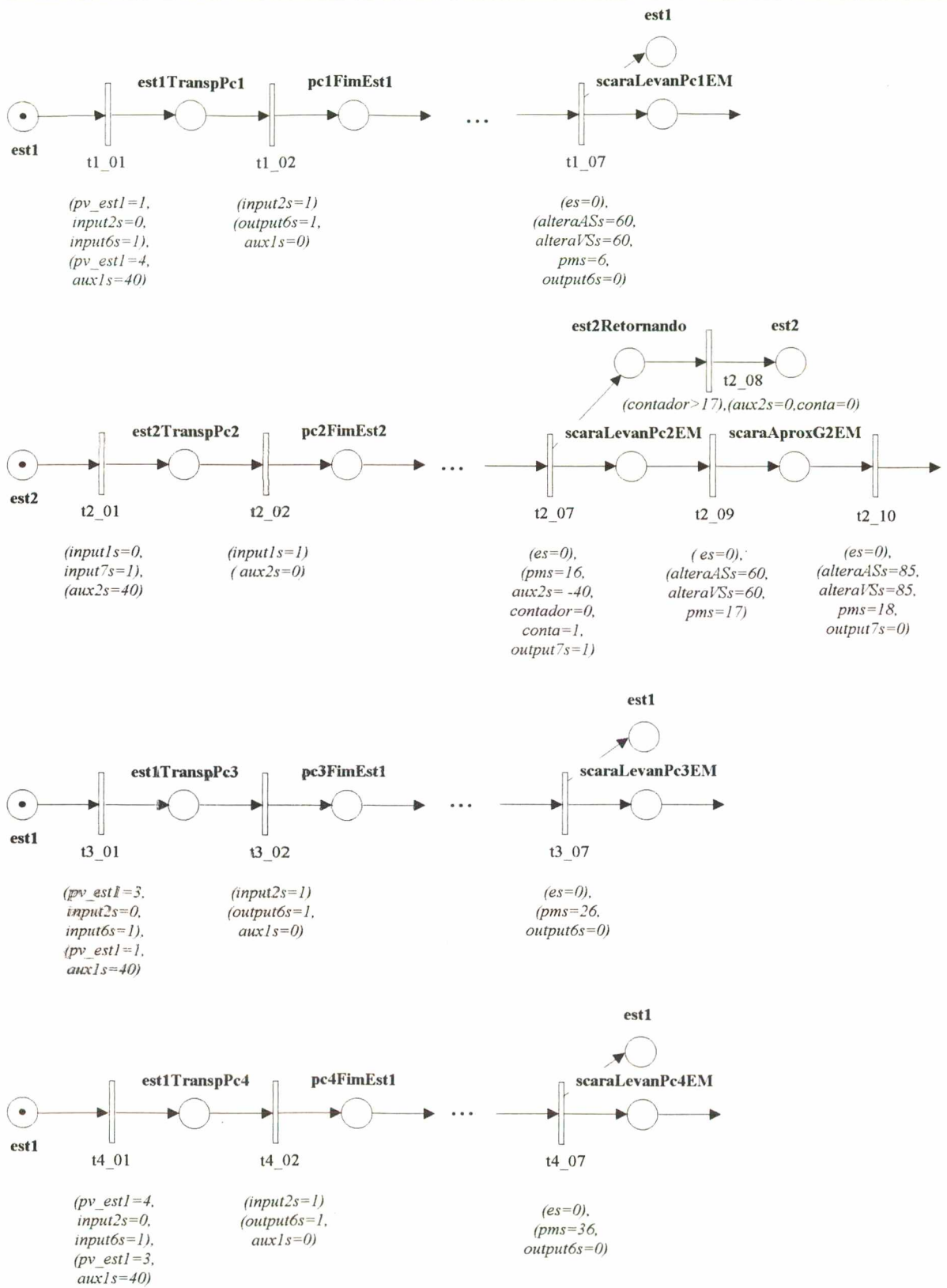


Figura 7.4 - Transições na RPI da FMC2 que fazem a comunicação com gerente FMC1

Pode-se observar pelo desenho das transições que as entradas e saídas 6 são utilizadas para estabelecer o sincronismo através da esteira 1, enquanto que a entrada e saída 7, para o sincronismo por meio da esteira 2.

Veja, por exemplo, como é feita a comunicação entre os gerentes durante o envio da peça 1 da célula de fabricação para a de montagem:

1°. O gerente FMC1 (figura 7.3) informa para o gerente FMC2 que existe uma peça depositada no início da esteira 1 (*output6*=1 - T1_29).

2°. O gerente FMC2 (figura 7.4) ao receber esta informação (*input6s*=1 - T1_01), deve iniciar o movimento da esteira 1 (*aux1s=40* - T1_01), desde que não haja peça no fim da esteira, que seria o caso da peça anterior ainda não ter sido retirada. Nesta situação (de não haver peça no fim da esteira 1), o sensor óptico 2 tem que estar desativado(*input2s*=0 - T1_01).

3°. Quando a peça chega no fim da esteira é ativado o sensor óptico 2 (*input2s*=1 - T1_02). Com isso o gerente FMC2 interrompe o movimento da esteira (aux1s=0 - T1_02) e avisa para o gerente FMC1 que a esteira já está pronta para receber uma nova peça (*output6s*=1 - T1_02). Logo depois, na transição T1_07, ele zera *output6s* para poder estabelecer a comunicação para o transporte da próxima peça.

4°. O gerente FMC1, quando recebe o sinal que a esteira está disponível (*input6*=1 - T1_30), coloca *output6* em zero (T1_30), que coloca uma ficha no lugar **est1**, indicando realmente que a esteira 1 está disponível para transportar a próxima peça.

7.2 - PROBLEMAS E SOLUCÕES

Alguns problemas que ocorreram durante a integração das células, alguns deles já mencionados em capítulos anteriores, foram:

(a) a necessidade de separar (resistor de 1KΩ) os sinais utilizados pelos sensores dos sinais de integração (seção 5.3.2);

(b) todos os sinais terra (GND) dos três controladores têm que ser colocados em comum (curto-circuitados), para que os demais sinais operem corretamente (capítulo 6); e

(c) necessidade de se ter um *pallet* para transportar a peça do tipo 2. Este *pallet* também é utilizado para corrigir os erros causados pela pouca precisão e repetibilidade do robô XR4. A maneira como isto é feito é muito semelhante ao posicionamento da peça 2 na peça base (seção 6.2.2), isto é: o robô não deve depositar a peça no *pallet*, mas sim soltá-la, de uma altura 'h', próxima à base do *pallet*. Como o *pallet* é formado por duas pequenas hastes cilíndricas, a peça, mesmo lançada de uma posição incorreta, acaba se ajustando entre as hastes, corrigindo o erro trazido pelo robô (figuras 7.5 e 7.6).

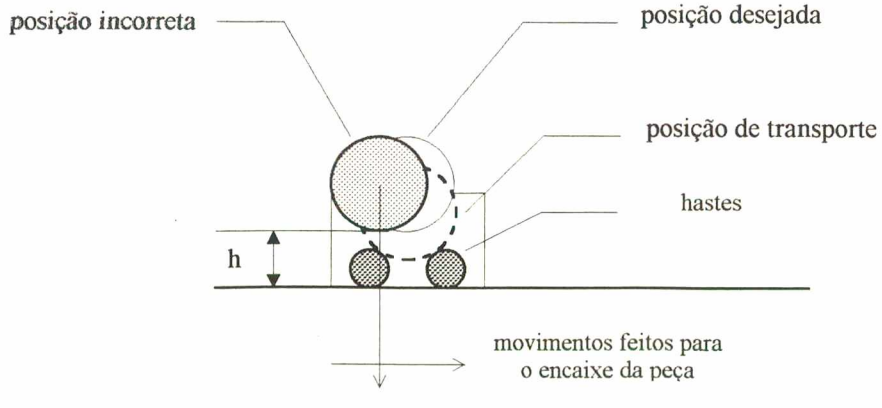


Figura 7.5 - Transporte da peça 2

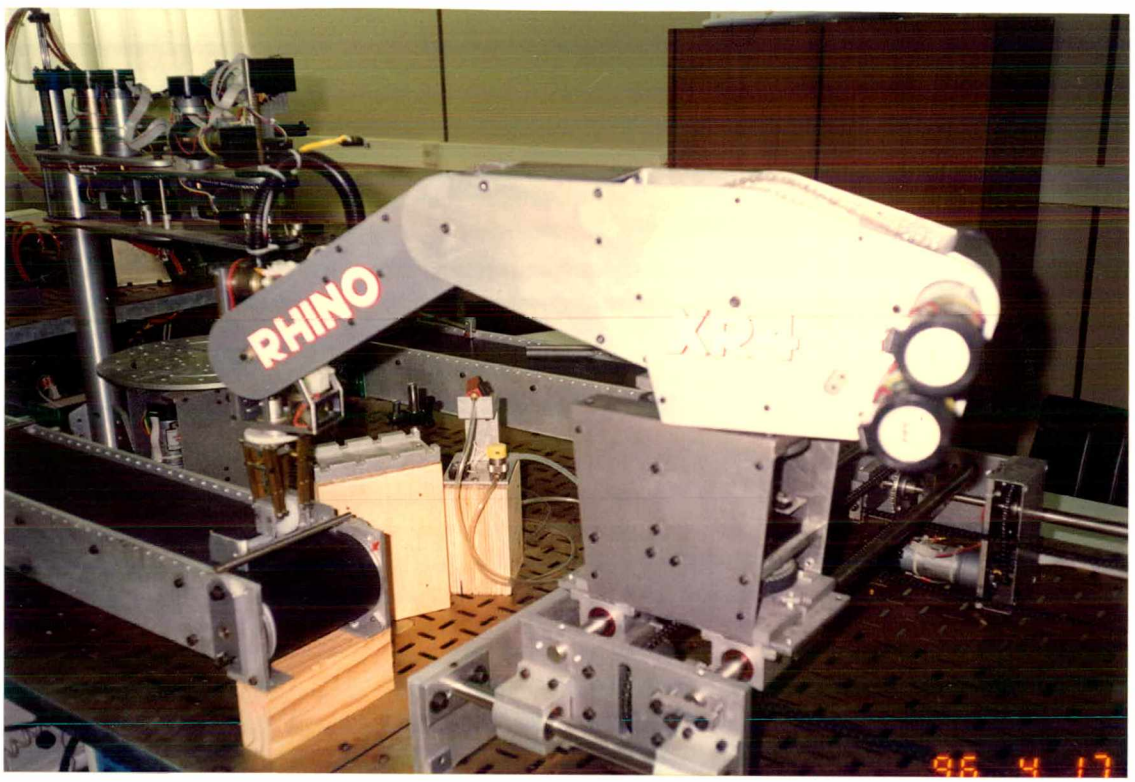


Figura 7.6 - Foto do XR4 (FMC1) depositando a peça 2 no *pallet* da esteira 2

(d) para corrigir os erros de posicionamento das peças na esteira 1 foi desenvolvido um dispositivo chamado *direcionador*. Com o direcionador as peças, na medida em que se movem sobre a esteira, são guiadas ou direcionadas pelo dispositivo (ver figura 7.7). Desta maneira, todas são sempre colocadas numa mesma posição, eliminando assim as indesejadas variações de posicionamento trazidas pelo robô da célula de fabricação.

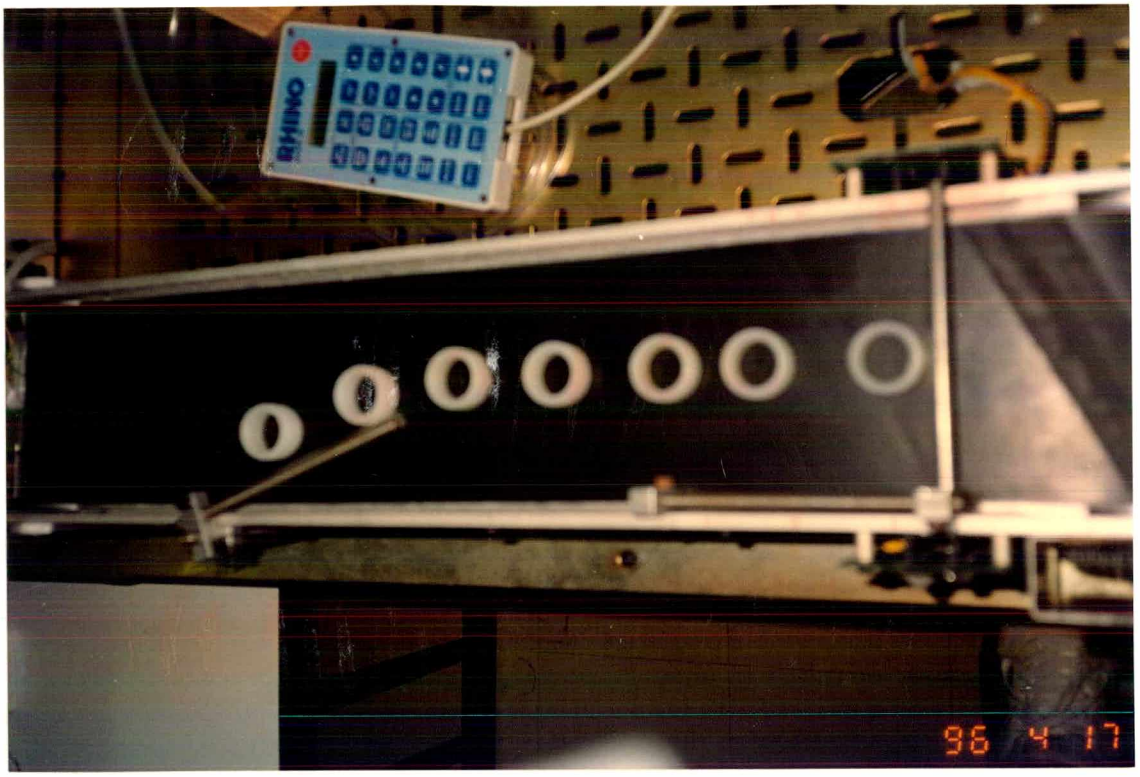


Figura 7.7 - Foto da esteira 1 com o direcionador

7.3 - RESUMO

Este capítulo procurou mostrar como foi feita a integração das células flexíveis de montagem e de fabrição. Fisicamente esta integração é feita por um cabo que liga três entradas de um controlador a três saídas do outro, e vice-versa. Logicamente a integração é feita através da troca de informações

entre os gerentes, especificada através das redes de Petri de cada célula.

Percebeu-se que, apesar dos erros causados pelos robôs, foi possível estabelecer uma perfeita integração entre as células. Para isto, foi necessário fazer com que os erros produzidos pela FMC1 não fossem transmitidos à FMC2, através do desenvolvimento de dispositivos como o *pallet* transportador (na esteira 2) e o direcionador (na esteira 1). Na figura 7.8, uma foto mostra o XR4 da FMC2 depositando o produto copletamente montado na FMC3, e na figura 7.9 é dado uma visão geral do sistema produtivo desenvolvido.

Figura 7.8 - Foto da integração FMC2-FMC3

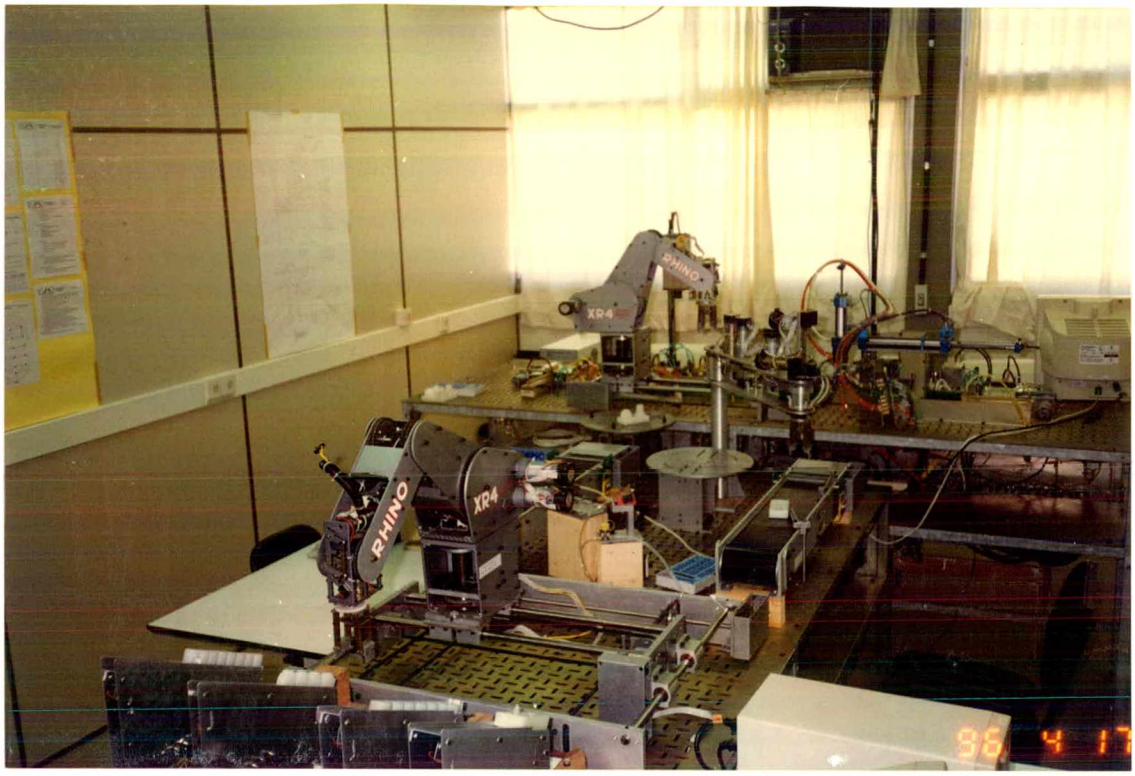


Figura 7.9 - Foto: visão geral do sistema produtivo proposto

8

CONCLUSÕES E FUTUROS TRABALHOS

Os objetivos propostos foram alcançados. Antes existia uma série de dispositivos que operavam isoladamente e agora, após o término do trabalho, existe um sistema de produção completamente automatizado, semelhante ao encontrado em grandes indústrias.

O trabalho não contribuiu no sentido da implementação de um CIM, nem de um FMS, porém motiva os leitores a caminhar neste sentido, haja visto que lida com assuntos referentes à implementação de células flexíveis de manufatura.

Este sistema foi formado pela integração de uma célula flexível de fabricação (FMC1) e uma célula flexível de montagem (FMC2). A FMC1 faz a simulação da fabricação de peças prismáticas e/ou rotacionais e a inspeção das mesmas. A FMC2 realiza a montagem dessas peças para gerar um produto final.

Uma das características marcantes durante o desenvolvimento do trabalho foi a verificação que realmente existe uma separação entre o que se vê na teoria e o que se tem na prática. Erros e problemas, a maioria deles causados por características de fabricação dos equipamentos, que em teoria não seriam considerados, estiveram a todo momento presentes durante o desenvolvimento.

À medida que estes problemas apareciam, novas soluções iam sendo encontradas, como por exemplo: (a) a necessidade de se ter que trabalhar com baixa aceleração e velocidade dos robôs, (b) a utilização de um *pallet* transportador, (c) a utilização de posicionadores e de um direcionador, (d) o ponto de conformação,

(e) os ampliadores para as garras dos robôs, e (f) o sensor para o *buffer* de entrada da peça 3.

Isto não quer dizer que a prática é mais importante que a teoria, ou vice-versa, muito pelo contrário, ambas são igualmente importantes e se complementam, de forma que o ensino somente pode estar perfeitamente relacionado com a realidade social se ambas estiverem em harmonia e interligadas.

Os *softwares* de gerenciamento das células (gerentes FMC) foram desenvolvidos para satisfazer tanto às funções de atendimento à célula (controle dos movimentos dos robôs, das esteiras, das estações de montagem, etc) quanto ao operador da mesma.

No gerente da célula o usuário pode implementar as FAC da sua própria configuração através de uma rede de Petri interpretada, e pode acessar as FAU através das opções nos menus.

A integração física das células foi feita por meio de um cabo formado por três pares de fios, ligando três entradas de um controlador a três saídas do outro, e vice-vesa. Para a integração lógica, as trocas de informações entre os gerentes é especificada pelo usuário através das RPIs das células.

Dentre as principais contribuições do presente trabalho, pode-se citar:

(a) Integração de uma FMC que simula a fabricação de peças por meio da usinagem, e o desenvolvimento do seu *software* de gerenciamento (gerente FMC1).

(b) Integração de uma FMC que realiza a montagem de peças, e o desenvolvimento do seu respectivo *software* de gerenciamento (gerente FMC2).

(c) Integração física e lógica de duas células flexiveis de manufatura, formando um pequeno sistema produtivo automatizado.

(d) Utilização de redes de Petri no gerenciamento e na integração de células flexiveis de manufatura.

(e) Desenvolvimento dos *drivers* de comunicação dos controladores MARK-IV. Com isto, qualquer aplicação feita em linguagem C++ poderá utilizar esses módulos.

(f) As disciplinas relacionadas à área de automação industrial da UFSC poderão se valer do sistema desenvolvido para suas aulas.

Ainda não existem conceitos padronizados para CIM, FMS e FMC, e por isso, baseando-se em definições feitas por profissionais da área, foram desenvolvidas conceituações próprias para esses sistemas. Acredita-se que esta também possa ser uma contribuição do trabalho desenvolvido.

A não existência de máquinas CNC e a não utilização de uma rede de comunicação local são as maiores simplificações do trabalho, sendo esta última uma conseqüência da primeira, pois, como não haviam máquinas, não havia a necessidade de se utilizar uma rede de comunicação. Os dispositivos foram perfeitamente integrados através dos controladores MARK-IV dos robôs.

Outras limitações dizem respeito à ausência de estoques intermediários e de peças para retrabalho, limitações estas que poderão ser consideradas no desenvolvimento de um trabalho futuro.

8.1 - TRABALHOS FUTUROS

Uma sugestão à continuidade deste trabalho seria integrar o sistema produtivo desenvolvido a outros dispositivos, como um banco de dados, um sistema de planejamento e controle da produção, sistemas de engenharia, com o objetivo de transformá-lo ou aproximá-lo de um sistema flexível de manufatura acadêmico.

Para fazer tal integração, deverá ser adquirido mais um computador (para rodar o gerente FMS), uma rede de comunicação local e algumas placas multiseriais. Os gerentes das FMCs desenvolvidos também deverão ser expandidos, para permitir a interface: gerente FMS↔gerentes FMC.

Algumas estratégias de produção como *just-in-time*, controle da qualidade total e tecnologia de grupo, junto com técnicas como inteligência artificial e redes neurais, poderão ser utilizadas neste futuro trabalho.

Uma outra sugestão seria considerar a possibilidade de haver quebras de máquinas, peças para retrabalho e a utilização de estoques intermediários.

Devido a baixa velocidade e aceleração que o sistema opera, causado principalmente pelo fato que o gerente FMC tem que atender tanto ao usuário quanto a própria célula (FAU e FAC, respectivamente), uma outra proposta para continuidade do trabalho seria aumentar esta velocidade de operação, modificando a maneira como é feito o atendimento ao usuário, através, por exemplo, de interrupção e/ou via arquivo texto.

BIBLIOGRAFIA

Aletan, S., The components of a successful CIM implementation, *Industrial Engineering*, 23(11), 20-22, 1991.

Arizono, I., **Yamamoto**, A. e **Ohta**, H., Scheduling for minimizing total actual flow time by neural networks, *International Journal of Production Research*, 30(3), 503-511, 1992.

Black, J.T., The design of the factory with a future, *McGraw-Hill*, 1991.

Braga, M.C.L.F., Um gerente de célula flexivel de manufatura, *Dissertação de Mestrado*, UFSC, Setembro, 1993.

Chan, C. e **Wang**, H., Design and development of a stochastic high-level Petri net system for FMS performance evaluation, *International Journal of Production Research*, 31(10), 2415-2439, 1993.

Choi, B.W., **Kuo**, W. e **Jackman**, J.K., Petri net extensions for modelling and validating manufacturing systems, *International Journal of Production Research*, 32(8), 1819-1835, 1994.

Chu, C., Manufacturing cell formation by competitive learning, *International Journal of Production Research*, 31(4), 829-843, 1993.

Cossins, R. e **Ferreira**, P., Celeritas: a coloured Petri net approach to simulation and control of flexible manufacturing systems, *International Journal of Production Research*, 30(8), 1925-1956, 1992.

Culbreth, C.T. e **Pollpeter**, D.L., A flexible manufacturing cell for furniture part production, *Industrial Engineering*, 20(11), 28-34, 1988.

Di Leva, A., **Giolito**, P. e **Vernadat**, F., M*-object: an object-oriented database design methodology for CIM information systems, *Control Engineering Practice*, 1(1), 183-187, 1993.

Graefe, U., **Pardasanim**, A. e **Chan**, A.W., Conceptual architecture of an object library for design, control and simulation of a manufacturing enterprise, *Control Engineering Practice*, 1(1), 141-146, 1993.

Hedin, S.R., **Malhotra**, M.K. e **Philipoom**, P.R., Shop floor control and tool loading policies in flexible manufacturing systems, *International Journal of Production Research*, 32(11), 2495-2512, 1994.

Hill, R.C., Computer integrated manufacturing platforms: DOS or UNIX?, *Industrial Engineering*, 24(11), 36-37, 1992.

Hohner, G., Local area networks links manufacturing cells in modular growth design, *Industrial Engineering*, 21(7), 23-28, 1989.

Huang, H. e **Chang**, P., Specifications, modelling and control of a flexible manufacturing cell, *International Journal of Production Research*, 30(11), 2515-2543, 1992.

Kaelli, K.J., A company-wide perspective to identify, evaluate, and rank the potencial for CIM, *Industrial Engineering*, 22(7), 23-26, 1990.

Kaltwasser, C., Know how to choose the right CIM systems integrator, *Industrial Engineering*, 22(7), 27-29, 1990.

Kellso, J.R, CIM in action: Microelectronics manufacturer charts course toward true systems integration, *Industrial Engineering*, 21(7), 19-22, 1989.

King, L., Today and tomorrow: speaking the language of the factory, *Industrial Engineering*, 23(11), 18-19, 1991.

Knight, D.O. e **Wall**, M.L., Using group technology for improving communication and coordination among teams of workers in manufacturing cells, *Industrial Engineering*, 21(1), 32-34, 1989.

Kochikar, V.P. e **Narendran**, T.T., A framework for assessing the flexibility of manufacturing systems, *International Journal of Production Research*, 30(12), 2873-2895, 1992.

Kochikar, V.P. e **Narendran**, T.T., On using abstract models for analysis of flexible manufacturing systems, *International Journal of Production Research*, 32(10), 2303-2322, 1994.

Kusiak, A., Intelligent Manufacturing Systems, *Prentice-Hall International Editions*, 1990.

Lopes, P.F., CIM II: The integrated manufacturing enterprise, *Industrial Engineering*, 24(11), 43-45, 1992.

Maccartthy, B.L. e **Liu**, J., A new classification scheme for flexible manufacturing Systems, *International Journal of Production Research*, 31(2), 299-309, 1993.

Nagarkar, S. e **Bennett**, D., Flexible manufacturing system lets small manufacturer of mainframes compete with giants, *Industrial Engineering*, 20(11), 42-46, 1988.

Nyman, L.R., Making manufacturing cells work, *Society of Manufacturing Engineers*, 1992.

Ohsen, C. von, Implementing CIM in a small company, *Industrial Engineering*, 24(11), 39-42, 1992.

Perry, G., Programação orientada para objetos com turbo C++, *Berkeley Brasil Editora*, Rio de Janeiro,1994.

Rabelo, L.C. e **Alptekin**, S., Integrating scheduling and control fuctions in computer-integrated manufacturing using artificial intelligence, *Computers and Industrial Engineering*, 14, 101-106, 1989.

Rao, H.A. e **Gu**, P., A multi-constraint neural network for the pragmatic design of cellular manufacturing systems, *International Journal of Production Research*, 33(4), 1049-1070, 1995.

Reddy, C.E., **Chetty**, O.V.K. e **Chaudhuri**, D., A Petri net based approach for analysing tool management issues in FMS, *International Journal of Production Research*, 30(6), 1427-1446, 1992.

Rehg, J.A., Computer-integrated manufacturing, *Prentice Hall*, 1994.

Rembold, U., **Nnaji**, B.O. e **Storr**, A., Computer Integrated Manufacturing and Engineering, *Addison-Wesley Publishing Company*, 1993.

Rhino Robots, XR Accessories: Owner's manual, *Manual Part Number FG-2952*, 1989.

Rhino Robots, The XR3, XR4, and SCARA Manual, *Manual Part Number FG-3056*, 1991.

Rhino Robots, Mark-IV 8 axis controller - Owner's manual, *Manual Part Number 65-060-6023*, 1989.

Sonntag, V., Flexible manufacturing...from a different perspective, *Industrial Engineering*, 22(11), 58-61, 1990.

Teng, S. e **Zhang**, J., A Petri net-based decomposition approach in modelling of manufacturing systems, *International Journal of Production Research*, 31(6), 1423-1439, 1993.

Udo, G.J., Neural networks applications in manufacturing processes, *Computers and Industrial Engineering*, 23(1-4), 94-100, 1992.

Vendrameto, O., Bases de conhecimento para a automação da manufatura, *Tese de Doutorado*, USP, 1994.

Venugopal, V. e **Narendran**, T.T., Machine-cell formation through neural networks models, *International Journal of Production Research*, 32(9), 2105-2116, 1994.

Vujosevic, R., Visual interactive simulation and artificial intelligence in design of flexible manufacturing systems, *International Journal of Production Research*, 32(8), 1955-1971, 1994.

Welke, H.A. e **Overbeeke**, J., Cellular manufacturing: a good technique for implementing just-in-time and total quality control, *Industrial Engineering*, 20(11), 36-41, 1988.

Yim, D. e **Linn**, R.J., Push and pull rules for dispatching automated guided vehicles in a flexible manufacturing system, *International Journal of Production Research*, 31(1), 43-57, 1993.

Young, R.E., **Greef**, A. e **O'Grady**, P., An artificial intelligence-based constraint network system for concurrent engineering, *International Journal of Production Research*, 30(7),1715-1735, 1992.

Zaremba, M.B., Information control for modern manufacturing systems; INCOM'92 editorial, *Control Engineering Practice*, 1(1), 121-126, 1993.

Zhang, H.C. e **Huang**, S.H., Applications of neural networks in manufacturing: a state-of-the-art survey, *International Journal of Production Research*, 33(3), 705-728, 1995.

ANEXO I: AS PRINCIPAIS FUNÇÕES DOS PROGRAMAS

Classe: *Rede de Petri Interpretada*:

```
void       cria(char *nome=NULL, ListaDeVariaveis *le=NULL, long
           int *t=NULL, int prioridade=0, Escalonador *esc=NULL)
int        lerRede(char *nome_Rede=NULL, int mostrarErro=NAO,
           int ordenando=NAO)
char       *nome(void)
Lugar      *criaLugar(char *nomeDoLugar=NULL,
           unsigned numDeFichas=0)
Transicao  *criaTransicao(char *nomeDaTrans=NULL)
Variavel   *criaVariavel(char *nomeDaVar=NULL, int valor=0)
Lugar      *existeLugar(char *nomeDoLugar)
Transicao  *existeTransicao(char *nomeDaTrans)
Variavel   *existeVariavelInterna(char *nomeDaVar)
Variavel   *existeVariavelExterna(char *nomeDaVar)
void       prioridade(int novaPrioridade)
Transicao  *joga(int aleat=NAO)
void       libera(void)
int        retiraLugar(char *nomeDoLugar)
int        retiraTransicao(char *nomeDaTrans)
int        retiraVariavel(char *nomeDaVar)
int        salva(char *nomeDaRede=NULL)
int        salvaSintaxeSp(char opcao, char *nomeDaRede=NULL)
void       mostraLugares(void)
void       mostraTransicoes(void)
void       mostraVariaveisInternas(void)
void       mostraVariaveisExternas(void)
int        numeroDeLugares(void)
Lugar      *retornaLugar(int posicao)
int        numeroDeTransicoes(void)
Transicao  *retornaTransicao(int posicao)
int        numeroDeTransHab(void)
Transicao  *retornaTransHab(int posicao)
int        numeroDeTransTemp(void)
Transicao  *retornaTransTemp(int posicao)
int        numeroDeVariaveisInternas(void)
Variavel   *retornaVariavelInterna(int posicao)
int        numeroDeVariaveisExternas(void)
Variavel   *retornaVariavelExterna(int posicao)
ListaDeVariaveis *retornaListaDeVariaveisExternas(void)
```

Classe: *Driver* de Comunicação SCARA:

```
int      envCom(unsigned char *comand)
int      envComRecTexto(unsigned char *comand)
int      envComRecInt(unsigned char *comand)
int      envComRecFloat(unsigned char *comand)
int      envComRecByte(unsigned char *comand)
int      configurate(int porta)
int      controlToHost()
int      controlToPendant()
int      goToSystemHardHome(int dG, int dH)
int      goToHardHome()
int      goToSoftHome()
int      gripperStatus(int *gs)
int      openGripper()
int      closeGripper()
int      readActualPosition(char *motor, int *ap)
int      readSoftHomePosition(char *motor, int *shp)
int      readSystemAcceleration(int *sa)
int      setOutputBit(int pos, int s)
int      setOutputPort(int d)
int      readInputBit(int pos, int *ib)
int      readInputPort(int *ip)
int      readOutputPort(int *op)
int      readAuxPortLevelDirection(int d, int *apld)
int      finalPosition(int posB,int posC,int posD,int posE)
int      moveGH(int posG, int posH)
int      moveMotor(char *motor, int pos)
int      hardHomeOnLimitSwitch(char *motor, int d=2)
int      clearActualPosition(char *motor)
int      setDestinationPositionABS(char *motor, int d)
int      setCoordinatePosition(int d)
int      setSystemAcceleration(int d)
int      setSystemVelocity(int d)
int      setMotorMode(char *motor, int d)
int      setMotorVelocity(char *motor, int d)
int      setAuxiliarPort(int AuxPort, int level_direction)
int      setSoftHome()
void     setHome(int gr,int posB,int posC,int posD,int posE)
int      setDestPositionABSonHome()
int      stopAll()
int      systemStatus(int *ss)
int      motorStatus(int *ms)
int      systemError(int *se)
int      robotStatus(int *rs)
int      executingMove(char *motor, int *em)
int      startCoordinateMove()
int      startIndependentMove()
```

Classe: *Driver* de Comunicação XR4:

```
int     envCom(unsigned char *comand)
int     envComRecTexto(unsigned char *comand)
int     envComRecInt(unsigned char *comand)
int     envComRecFloat(unsigned char *comand)
int     envComRecByte(unsigned char *comand)
int     configurate(int porta)
int     controlToHost()
int     controlToPendant()
int     goToSystemHardHome(int dG, int dH)
int     goToHardHome()
int     goToSoftHome()
int     gripperStatus(int *gs)
int     openGripper()
int     closeGripper()
int     readActualPosition(char *motor, int *ap)
int     readSoftHomePosition(char *motor, int *shp)
int     readSystemAcceleration(int *sa)
int     setOutputBit(int pos, int s)
int     setOutputPort(int d)
int     readInputBit(int pos, int *ib)
int     readInputPort(int *ip)
int     readOutputPort(int *op)
int     readAuxPortLevelDirection(int d, int *apld)
int     finalPosition(int posB,int posC,int posD,int posE,
        int posF)
int     moveGH(int posG, int posH)
int     moveMotor(char *motor, int pos)
int     hardHomeOnLimitSwitch(char *motor, int d=2
int     clearActualPosition(char *motor)
int     setDestinationPositionABS(char *motor, int d
int     setCoordinatePosition(int d)
int     setSystemAcceleration(int d)
int     setSystemVelocity(int d)
int     setMotorMode(char *motor, int d)
int     setMotorVelocity(char *motor, int d)
int     setAuxiliarPort(int AuxPort, int level_direction)
int     setSoftHome()
void    setHome(int gr,int posB,int posC,int posD,int posE,
        int posF)
int     setDestPositionABSonHome()
int     stopAll()
int     systemStatus(int *ss)
int     systemError(int *se)
int     robotStatus(int *rs)
int     executingMove(char *motor, int *em)
```

```
int       startCoordinateMove()
int       startIndependentMove()
```

Classe: *INWIN*:

```
int     create(char tip,int bor, int fbc, int ftc, int obc,
        int otc, int sbc,int stc, int l, int t, char *opc[],
        int w, int numop)
int     alterLine(int pos, char *str)
int     alterFrameColor(int newfbc, int newftc)
int     alterOptionColor(int pos, int newobc, int newotc)
int     show()
void    hide()
int     selected()
int     active()
void    freeWin()
int     option()
int     option(int tecla)
```

Classe: *Dispositivo Máquina*:

```
void      cria(char *nom=NULL, char *operac=NULL, float pq=0)
void      trabalha(char *nomePeca=NULL, unsigned tt)
void      libera(void)
void      atualizaTempo(void)
void      seMostra(void)
unsigned  estadoAtual(void)
void      alteraProbQuebra(float npq)
void      alteraTempoTrabalho(unsigned ntt)
```

ANEXO II: OS PRINCIPAIS COMANDOS UTILIZADOS

Comandos do sistema:

SA	Ler *status* dos motores
SC	Ler a configuração do sistema
SE	Ler a lista de erros
SS	Ler *status* do sistema

Comandos de configuração:

CC	Passar para coordenada XYZ
CG	Habilitar/Desabilitar Garra
CM	Atribuir o modo do motor
CR	Atribuir o tipo de robô

Comandos de leitura do motor:

GS	Ler o status da garra
PA	Ler posição atual
RL	Ler chaves de fim-de-curso
XR	Ler o nível e a direção das portas auxiliares

Comandos de atribuição do motor:

AC	Limpar a posição atual do motor
AS	Atribuir a aceleração do sistema
GC	Fechar a garra
GO	Abrir a garra
HA	Ir para a posição Hard-Home
MA	Parar todos os motores
MC	Iniciar motores, coordenados
MI	Iniciar motores, independentes
PD	Atribuir a posição destino, absoluta
PR	Atribuir a posição destino, relativa
VG	Atribuir a velocidade do sistema
VS	Atribuir a velocidade do motor
XS	Atribuir o nível e a direção da porta auxiliar

Comandos do *teach pendant*:

TE	Habilita/Desabilita o teach pendant para mover os motores
TH	Passar o controle para o computador hospedeiro
TX	Passar o controle para o *teach pendant*

Comandos de entrada e saída:

IB	Ler bit de entrada
IP	Ler porta de entrada
OB	Atribuir bit de saída
OP	Atribuir porta de saída
OR	Ler porta de saída

ANEXO III: OS ARQUIVOS DAS RPIS DESENVOLVIDAS

Arquivo da Rede da Célula de Fabricação:

```
Rede FMC1;
Nodos
proxMesa1, rbAproxBuf1, rbAproxGBuf1, rbSegurPcb1, rbAfastGBuf1,
rbAproxPcb1Torno, rbAproxGPcb1Torno, pcb1Torno, rbAfastG1Torno, tornoUsinPc1,
pc1Pronta, rbAprox1Torno, rbAproxG1Torno, rbSegurPc1, rbAfastGPc1Torno,
rbAproxPc1SV, rbAproxGPc1SV, pc1SV, rbAfastG1SV, SVInspec1, pc1Inspec,
rbAproxG1SV, rbSegurPci1, rbAfastGPc1SV, rbAproxPc1Est1, rbAproxGPc1Est1,
pc1Est1, rbAfastG1Est1, est1TranspPc1, rbAproxPc1BR, rbAproxGPc1BR, pc1BR,
rbAfastG1BR,

aproxMesa2, rbAproxBuf2, rbAproxGBuf2, rbSegurPcb2, rbAfastGBuf2,
rbAproxPcb2Torno, rbAproxGPcb2Torno, pcb2Torno, rbAfastG2Torno, tornoUsinPc2,
pc2Pronta, rbAprox2Torno, rbAproxG2Torno, rbSegurPc2, rbAfastGPc2Torno,
rbAproxPc2SV, rbAproxGPc2SV, pc2SV, rbAfastG2SV, SVInspec2, pc2Inspec,
rbAproxG2SV, rbSegurPci2, rbAfastGPc2SV, rbAproxPc2Est2, rbAproxGPc2Est2,
pc2Est2, rbAfastG2Est2, est2TranspPc2, rbAproxPc2BR, rbAproxGPc2BR, pc2BR,
rbAfastG2BR,

aproxMesa3, rbAproxBuf3, rbAproxGBuf3, rbSegurPcb3, rbAfastGBuf3,
rbAproxPcb3Torno, rbAproxGPcb3Torno, pcb3Torno, rbAfastG3Torno, tornoUsinPc3,
pc3Pronta, rbAprox3Torno, rbAproxG3Torno, rbSegurPc3, rbAfastGPc3Torno,
rbAproxPc3SV, rbAproxGPc3SV, pc3SV, rbAfastG3SV, SVInspec3, pc3Inspec,
rbAproxG3SV, rbSegurPci3, rbAfastGPc3SV, rbAproxPc3Est1, rbAproxGPc3Est1,
pc3Est1, rbAfastG3Est1, est1TranspPc3, rbAproxPc3BR, rbAproxGPc3BR, pc3BR,
rbAfastG3BR,

aproxMesa4, rbAproxBuf4, rbAproxGBuf4, rbSegurPcb4, rbAfastGBuf4,
rbAproxPcb4CU, rbAproxGPcb4CU, pcb4CU, rbAfastG4CU, cuUsinPc4, pc4Pronta,
rbAprox4CU, rbAproxG4CU, rbSegurPc4, rbAfastGPc4CU, rbAproxPc4SV,
rbAproxGPc4SV, pc4SV, rbAfastG4SV, SVInspec4, pc4Inspec, rbAproxG4SV,
rbSegurPci4, rbAfastGPc4SV, rbAproxPc4Est1, rbAproxGPc4Est1, pc4Est1,
rbAfastG4Est1, est1TranspPc4, rbAproxPc4BR, rbAproxGPc4BR, pc4BR,
rbAfastG4BR : Lugar;

torno, cu, robo, SV, est1, est2                                    : Lugar(1);

{peca 4 mais prioritaria}
t4_01, t4_02, t4_03, t4_04, t4_05, t4_06, t4_07, t4_08, t4_09, t4_10, t4_11,
t4_12, t4_13, t4_14, t4_15, t4_16, t4_17, t4_18, t4_19, t4_20, t4_21, t4_22,
t4_23, t4_24, t4_25, t4_26, t4_27, t4_28, t4_29, t4_30, t4_31, t4_32, t4_33,
t4_34, t4_35,

t3_01, t3_02, t3_03, t3_04, t3_05, t3_06, t3_07, t3_08, t3_09, t3_10, t3_11,
t3_12, t3_13, t3_14, t3_15, t3_16, t3_17, t3_18, t3_19, t3_20, t3_21, t3_22,
t3_23, t3_24, t3_25, t3_26, t3_27, t3_28, t3_29, t3_30, t3_31, t3_32, t3_33,
t3_34, t3_35,

t2_01, t2_02, t2_03, t2_04, t2_05, t2_06, t2_07, t2_08, t2_09, t2_10, t2_11,
t2_12, t2_13, t2_14, t2_15, t2_16, t2_17, t2_18, t2_19, t2_20, t2_21, t2_22,
t2_23, t2_24, t2_25, t2_26, t2_27, t2_28, t2_29, t2_30, t2_31, t2_32, t2_33,
t2_34, t2_35,

{peca 1, menos prioritaria}
```

```
t1_01, t1_02, t1_03, t1_04, t1_05, t1_06, t1_07, t1_08, t1_09, t1_10, t1_11,
t1_12, t1_13, t1_14, t1_15, t1_16, t1_17, t1_18, t1_19, t1_20, t1_21, t1_22,
t1_23, t1_24, t1_25, t1_26, t1_27, t1_28, t1_29, t1_30, t1_31, t1_32, t1_33,
t1_34, t1_35                                                    :Transicao;
```

Estrutura

```
t1_01:(torno,robo),(aproxMesa1_1);
t1_02:(aproxMesa1),(rbAproxBuf1);
t1_03:(rbAproxBuf1),(rbAproxGBuf1);
t1_04:(rbAproxGBuf1),(rbSegurPcb1);
t1_05:(rbSegurPcb1),(rbAfastGBuf1);
t1_06:(rbAfastGBuf1),(rbAproxPcb1Torno);
t1_07:(rbAproxPcb1Torno),(rbAproxGPcb1Torno);
t1_08:(rbAproxGPcb1Torno),(pcb1Torno);
t1_09:(pcb1Torno),(rbAfastG1Torno);
t1_10:(rbAfastG1Torno),(robo,tornoUsinPc1);
t1_11:(tornoUsinPc1),(pc1Pronta);
t1_12:(pc1Pronta,SV,robo),(rbAprox1Torno);
t1_13:(rbAprox1Torno),(rbAproxG1Torno);
t1_14:(rbAproxG1Torno),(rbSegurPc1);
t1_15:(rbSegurPc1),(rbAfastGPc1Torno);
t1_16:(rbAfastGPc1Torno),(rbAproxPc1SV,torno);
t1_17:(rbAproxPc1SV),(rbAproxGPc1SV);
t1_18:(rbAproxGPc1SV),(pc1SV);
t1_19:(pc1SV),(rbAfastG1SV);
t1_20:(rbAfastG1SV),(SVInspec1);
t1_21:(SVInspec1),(pc1Inspec);
t1_22:(pc1Inspec),(rbAproxG1SV);
t1_23:(rbAproxG1SV),(rbSegurPci1);
t1_24:(rbSegurPci1),(rbAfastGPc1SV);
t1_25:(rbAfastGPc1SV,est1),(rbAproxPc1Est1,SV);
t1_26:(rbAproxPc1Est1),(rbAproxGPc1Est1);
t1_27:(rbAproxGPc1Est1),(pc1Est1);
t1_28:(pc1Est1),(rbAfastG1Est1);
t1_29:(rbAfastG1Est1),(robo,est1TranspPc1);
t1_30:(est1TranspPc1),(est1);
t1_31:(rbAfastGPc1SV),(rbAproxPc1BR,SV);
t1_32:(rbAproxPc1BR),(rbAproxGPc1BR);
t1_33:(rbAproxGPc1BR),(pc1BR);
t1_34:(pc1BR),(rbAfastG1BR);
t1_35:(rbAfastG1BR),(robo);

t2_01:(torno,robo),(aproxMesa2);
t2_02:(aproxMesa2),(rbAproxBuf2);
t2_03:(rbAproxBuf2),(rbAproxGBuf2);
t2_04:(rbAproxGBuf2),(rbSegurPcb2);
t2_05:(rbSegurPcb2),(rbAfastGBuf2);
t2_06:(rbAfastGBuf2),(rbAproxPcb2Torno);
t2_07:(rbAproxPcb2Torno),(rbAproxGPcb2Torno);
t2_08:(rbAproxGPcb2Torno),(pcb2Torno);
t2_09:(pcb2Torno),(rbAfastG2Torno);
t2_10:(rbAfastG2Torno),(robo,tornoUsinPc2); t2_11:(tornoUsinPc2),(pc2Pronta);
t2_12:(pc2Pronta,SV,robo),(rbAprox2Torno);
t2_13:(rbAprox2Torno),(rbAproxG2Torno); t2_14:(rbAproxG2Torno),(rbSegurPc2);
t2_15:(rbSegurPc2),(rbAfastGPc2Torno);
t2_16:(rbAfastGPc2Torno),(rbAproxPc2SV,torno);
t2_17:(rbAproxPc2SV),(rbAproxGPc2SV); t2_18:(rbAproxGPc2SV),(pc2SV);
t2_19:(pc2SV),(rbAfastG2SV);
t2_20:(rbAfastG2SV),(SVInspec2);
```

```
t2_21:(SVInspec2),(pc2Inspec);
t2_22:(pc2Inspec),(rbAproxG2SV);
t2_23:(rbAproxG2SV),(rbSegurPci2);
t2_24:(rbSegurPci2),(rbAfastGPc2SV);
t2_25:(rbAfastGPc2SV,est2),(rbAproxPc2Est2,SV);
t2_26:(rbAproxPc2Est2),(rbAproxGPc2Est2); t2_27:(rbAproxGPc2Est2),(pc2Est2);
t2_28:(pc2Est2),(rbAfastG2Est2); t2_29:(rbAfastG2Est2),(robo,est2TranspPc2);
t2_30:(est2TranspPc2),(est2);
t2_31:(rbAfastGPc2SV),(rbAproxPc2BR,SV);
t2_32:(rbAproxPc2BR),(rbAproxGPc2BR); t2_33:(rbAproxGPc2BR),(pc2BR);
t2_34:(pc2BR),(rbAfastG2BR);
t2_35:(rbAfastG2BR),(robo);

t3_01:(torno,robo),(aproxMesa3);
t3_02:(aproxMesa3),(rbAproxBuf3);
t3_03:(rbAproxBuf3),(rbAproxGBuf3);
t3_04:(rbAproxGBuf3),(rbSegurPcb3);
t3_05:(rbSegurPcb3),(rbAfastGBuf3);
t3_06:(rbAfastGBuf3),(rbAproxPcb3Torno);
t3_07:(rbAproxPcb3Torno),(rbAproxGPcb3Torno);
t3_08:(rbAproxGPcb3Torno),(pcb3Torno); t3_09:(pcb3Torno),(rbAfastG3Torno);
t3_10:(rbAfastG3Torno),(robo,tornoUsinPc3); t3_11:(tornoUsinPc3),(pc3Pronta);
t3_12:(pc3Pronta,SV,robo),(rbAprox3Torno);
t3_13:(rbAprox3Torno),(rbAproxG3Torno); t3_14:(rbAproxG3Torno),(rbSegurPc3);
t3_15:(rbSegurPc3),(rbAfastGPc3Torno);
t3_16:(rbAfastGPc3Torno),(rbAproxPc3SV,torno);
t3_17:(rbAproxPc3SV),(rbAproxGPc3SV); t3_18:(rbAproxGPc3SV),(pc3SV);
t3_19:(pc3SV),(rbAfastG3SV);
t3_20:(rbAfastG3SV),(SVInspec3);
t3_21:(SVInspec3),(pc3Inspec);
t3_22:(pc3Inspec),(rbAproxG3SV);
t3_23:(rbAproxG3SV),(rbSegurPci3);
t3_24:(rbSegurPci3),(rbAfastGPc3SV);
t3_25:(rbAfastGPc3SV,est1),(rbAproxPc3Est1,SV);
t3_26:(rbAproxPc3Est1),(rbAproxGPc3Est1); t3_27:(rbAproxGPc3Est1),(pc3Est1);
t3_28:(pc3Est1),(rbAfastG3Est1); t3_29:(rbAfastG3Est1),(robo,est1TranspPc3);
t3_30:(est1TranspPc3),(est1);
t3_31:(rbAfastGPc3SV),(rbAproxPc3BR,SV);
t3_32:(rbAproxPc3BR),(rbAproxGPc3BR); t3_33:(rbAproxGPc3BR),(pc3BR);
t3_34:(pc3BR),(rbAfastG3BR);
t3_35:(rbAfastG3BR),(robo);

t4_01:(cu,robo),(aproxMesa4);
t4_02:(aproxMesa4),(rbAproxBuf4);
t4_03:(rbAproxBuf4),(rbAproxGBuf4);
t4_04:(rbAproxGBuf4),(rbSegurPcb4);
t4_05:(rbSegurPcb4),(rbAfastGBuf4);
t4_06:(rbAfastGBuf4),(rbAproxPcb4CU); t4_07:(rbAproxPcb4CU),(rbAproxGPcb4CU);
t4_08:(rbAproxGPcb4CU),(pcb4CU); t4_09:(pcb4CU),(rbAfastG4CU);
t4_10:(rbAfastG4CU),(robo,cuUsinPc4); t4_11:(cuUsinPc4),(pc4Pronta);
t4_12:(pc4Pronta,SV,robo),(rbAprox4CU);
t4_13:(rbAprox4CU),(rbAproxG4CU); t4_14:(rbAproxG4CU),(rbSegurPc4);
t4_15:(rbSegurPc4),(rbAfastGPc4CU); t4_16:(rbAfastGPc4CU),(rbAproxPc4SV,cu);
t4_17:(rbAproxPc4SV),(rbAproxGPc4SV); t4_18:(rbAproxGPc4SV),(pc4SV);
t4_19:(pc4SV),(rbAfastG4SV);
t4_20:(rbAfastG4SV),(SVInspec4);
t4_21:(SVInspec4),(pc4Inspec);
t4_22:(pc4Inspec),(rbAproxG4SV);
t4_23:(rbAproxG4SV),(rbSegurPci4);
t4_24:(rbSegurPci4),(rbAfastGPc4SV);
```

```
t4_25:(rbAfastGPc4SV,est1),(rbAproxPc4Est1,SV);
t4_26:(rbAproxPc4Est1),(rbAproxGPc4Est1); t4_27:(rbAproxGPc4Est1),(pc4Est1);
t4_28:(pc4Est1),(rbAfastG4Est1); t4_29:(rbAfastG4Est1),(robo,est1TranspPc4);
t4_30:(est1TranspPc4),(est1);
t4_31:(rbAfastGPc4SV),(rbAproxPc4BR,SV);
t4_32:(rbAproxPc4BR),(rbAproxGPc4BR); t4_33:(rbAproxGPc4BR),(pc4BR);
t4_34:(pc4BR),(rbAfastG4BR);
t4_35:(rbAfastG4BR),(robo);
Fim

Variaveis
torno_pr                                                    :Interna(3);
cu_pr, pv {peca da vez}                                     :Interna(4);
usina1, usina2, usina3                                      :Externa(5);
usina4                                                      :Externa(20);
inspeciona1, inspeciona2, inspeciona3                       :Externa(2);
inspeciona4                                                 :Externa(8);

input1, input2, input3, input4, input5, input6, input7, input8,
output1,output2,output3,output4,output5, output6,output7,output8,
tu1, tu2, tu3, tu4,
ti1, ti2, ti3, ti4,
probPeca1, probPeca2, probPeca3, probPeca4,
pmr, er, eg,
alteraAS, alteraVS,
npb1, npb2, npb3, npb4,
npr1, npr2, npr3, npr4                                      :Externa;

Etiquetas
t1_01:(input1=1, torno_pr=1),(torno_pr=0,pmr=2);
t1_02:(er=0),(pmr=3);
t1_03:(er=0),(pmr=4);
t1_04:(er=0),(pmr=5);
t1_05:(er=0),(pmr=6);
t1_06:(er=0),(pmr=7);
t1_07:(er=0),(pmr=8,alteraAS=45,alteraVS=45);
t1_08:(er=0),(pmr=9,alteraAS=70,alteraVS=70);
t1_09:(er=0),(pmr=10);
t1_10:(er=0),(usina1=1,output1=1);
t1_11:(usina1=0),(output1=0);
t1_12:(pv=1),(pmr=11);
t1_13:(er=0),(pmr=13);
t1_14:(er=0),(pmr=14);
t1_15:(er=0),(pmr=15);
t1_16:(er=0),(pmr=16,alteraAS=45,alteraVS=45);
t1_17:(er=0),(pmr=17,alteraAS=70,alteraVS=70);
t1_18:(er=0),(pmr=18);
t1_19:(er=0),(pmr=19);
t1_20:(er=0),(inspeciona1=1,output3=1);
t1_21:(inspeciona1=0 ),(output3=0);
t1_22:(er=0),(pmr=20);
t1_23:(er=0),(pmr=21);
t1_24:(er=0),(pmr=22);
t1_25:(er=0,probPeca1>=25),( npb1+=1,pmr=23,pv=4,torno_pr=3);
t1_26:(er=0),(pmr=24,alteraAS=45,alteraVS=45);
t1_27:(er=0),(pmr=25,alteraAS=70,alteraVS=70);
t1_28:(er=0),(pmr=26);
t1_29:(er=0),(output6=1);
t1_30:(input6=1),(output6=0);
```

```
t1_31:(er=0,probPeca1<25),(npr1+=1,torno_pr=1,pmr=27);
t1_32:(er=0),(pmr=28,alteraAS=45,alteraVS=45);
t1_33:(er=0),(pmr=29,alteraAS=70,alteraVS=70);
t1_34:(er=0),(pmr=30);
t1_35:(er=0),();
t2_01:(input2=1, torno_pr=2),(torno_pr=0,pmr=32);
t2_02:(er=0),(pmr=33);
t2_03:(er=0),(pmr=34);
t2_04:(er=0),(pmr=35);
t2_05:(er=0),(pmr=36);
t2_06:(er=0),(pmr=37);
t2_07:(er=0),(pmr=38,alteraAS=45,alteraVS=45);
t2_08:(er=0),(pmr=39,alteraAS=70,alteraVS=70);
t2_09:(er=0),(pmr=40);
t2_10:(er=0),(usina2=1,output1=1);
t2_11:(usina2=0),(output1=0);
t2_12:(pv=2),(pmr=41);
t2_13:(er=0),(pmr=43);
t2_14:(er=0),(pmr=44);
t2_15:(er=0),(pmr=45);
t2_16:(er=0),(pmr=46,alteraAS=45,alteraVS=45);
t2_17:(er=0),(pmr=47);
t2_18:(er=0),(pmr=48,alteraAS=70,alteraVS=70);
t2_19:(er=0),(pmr=49);
t2_20:(er=0),(inspeciona2=1,output3=1);
t2_21:(inspeciona2=0),(output3=0);
t2_22:(er=0),(pmr=50);
t2_23:(er=0),(pmr=51);
t2_24:(er=0),(pmr=52);
t2_25:(er=0,probPeca2>=25),(npb2+=1,pmr=53,pv=1,torno_pr=1);
t2_26:(er=0),(pmr=54,alteraAS=45,alteraVS=45);
t2_27:(er=0),(pmr=55,alteraAS=70,alteraVS=70);
t2_28:(er=0),(pmr=56);
t2_29:(er=0),(output7=1);
t2_30:(input7=1),(output7=0);
t2_31:(er=0,probPeca2<25),(npr2+=1,torno_pr=2,pmr=57);
t2_32:(er=0),(pmr=58,alteraAS=45,alteraVS=45);
t2_33:(er=0),(pmr=59,alteraAS=70,alteraVS=70);
t2_34:(er=0),(pmr=60);
t2_35:(er=0),();
t3_01:(input3=1, torno_pr=3),(torno_pr=0,pmr=62);
t3_02:(er=0),(pmr=63);
t3_03:(er=0),(pmr=64);
t3_04:(er=0),(pmr=65);
t3_05:(er=0),(pmr=66);
t3_06:(er=0),(pmr=67);
t3_07:(er=0),(pmr=68,alteraAS=45,alteraVS=45);
t3_08:(er=0),(pmr=69,alteraAS=70,alteraVS=70);
t3_09:(er=0),(pmr=70);
t3_10:(er=0),(usina3=1,output1=1);
t3_11:(usina3=0),(output1=0);
t3_12:(pv=3),(pmr=71);
t3_13:(er=0),(pmr=73);
t3_14:(er=0),(pmr=74);
t3_15:(er=0),(pmr=75);
t3_16:(er=0),(pmr=76);
t3_17:(er=0),(pmr=77,alteraAS=45,alteraVS=45);
t3_18:(er=0),(pmr=78,alteraAS=70,alteraVS=70);
t3_19:(er=0),(pmr=79);
t3_20:(er=0),(inspeciona3=1,output3=1);
```

```
t3_21:(inspeciona3=0),(output3=0);
t3_22:(er=0),(pmr=80);
t3_23:(er=0),(pmr=81);
t3_24:(er=0),(pmr=82);
t3_25:(er=0,probPeca3>=25),(npb3+=1,pmr=83,pv=2,torno_pr=2);
t3_26:(er=0),(pmr=84,alteraAS=45,alteraVS=45);
t3_27:(er=0),(pmr=85,alteraAS=70,alteraVS=70);
t3_28:(er=0),(pmr=86);
t3_29:(er=0),(output6=1);
t3_30:(input6=1),(output6=0);
t3_31:(er=0,probPeca3<25),(npr3+=1,torno_pr=3,pmr=87);
t3_32:(er=0),(pmr=88,alteraAS=45,alteraVS=45);
t3_33:(er=0),(pmr=89,alteraAS=70,alteraVS=70);
t3_34:(er=0),(pmr=70);
t3_35:(er=0),();
t4_01:(input4=1,cu_pr=4),(cu_pr=0,pmr=92);
t4_02:(er=0),(pmr=93);
t4_03:(er=0),(pmr=94);
t4_04:(er=0),(pmr=95);
t4_05:(er=0),(pmr=96);
t4_06:(er=0),(pmr=97);
t4_07:(er=0),(pmr=98,alteraAS=45,alteraVS=45);
t4_08:(er=0),(pmr=99,alteraAS=70,alteraVS=70);
t4_09:(er=0),(pmr=100);
t4_10:(er=0),(usina4=1,output2=1);
t4_11:(usina4=0),(output2=0);
t4_12:(pv=4),(pmr=101,cu_pr=4);
t4_13:(er=0),(pmr=103);
t4_14:(er=0),(pmr=104);
t4_15:(er=0),(pmr=105);
t4_16:(er=0),(pmr=106);
t4_17:(er=0),(pmr=107,alteraAS=45,alteraVS=45);
t4_18:(er=0),(pmr=108,alteraAS=70,alteraVS=70);
t4_19:(er=0),(pmr=109);
t4_20:(er=0),(inspeciona4=1,output3=1);
t4_21:(inspeciona4=0),(output3=0);
t4_22:(er=0),(pmr=110);
t4_23:(er=0),(pmr=111);
t4_24:(er=0),(pmr=112);
t4_25:(er=0,probPeca4>=25),(npb4+=1,pmr=113,pv=3);
t4_26:(er=0),(pmr=114,alteraAS=45,alteraVS=45);
t4_27:(er=0),(pmr=115,alteraAS=70,alteraVS=70);
t4_28:(er=0),(pmr=116);
t4_29:(er=0),(output6=1);
t4_30:(input6=1),(output6=0);
t4_31:(er=0,probPeca4<25),(npr4+=1,pmr=117);
t4_32:(er=0),(pmr=118,alteraAS=45,alteraVS=45);
t4_33:(er=0),(pmr=119,alteraAS=70,alteraVS=70);
t4_34:(er=0),(pmr=120);
t4_35:(er=0),();

Fim_Rede.
```

Arquivo da Rede da Célula de Montagem:

```
Rede FMC2;

Nodos   {a est1 esta ligada em AUX 1 e a est2 em AUX 2}
est1TranspPc1,       pc1FimEst1,       scaraAprox1Est1,       caraAproxG1Est1,
scaraSegurPc1,   scaraAfastG1Est1,   scaraLevanPc1EM,   scaraAproxG1EM,pc1EM,
scaraAfastG1EM, scaraAprox1Centro1,

xr4AproxEM,    xr4AproxGEM,    r4SegurPeca,    xr4AfastGEM,    xr4LevanPcFMC3,
xr4AproxGFMC3, pecaFMC3, xr4afastGFMC3, xr4AproxCentro,

est2TranspPc2, pc2FimEst2, scaraAprox2Est2, scaraAproxG2Est2, scaraSegurPc2,
scaraAfastG2Est2,   scaraLevanPc2EM,   scaraAproxG2EM,   pc2EM,   pc2Montada,
scaraAfastG2EM, scaraAprox2Centro2, est2Retornando,

est1TranspPc3, pc3FimEst1, scaraAprox3Est1, scaraAproxG3Est1, scaraSegurPc3,
scaraAfastG3Est1,   scaraLevanPc3EM,   scaraAproxG3EM,   pc3EM,   pc3Montada,
scaraAfastG3EM, scaraAprox3Centro3,

est1TranspPc4, pc4FimEst1, scaraAprox4Est1, scaraAproxG4Est1, scaraSegurPc4,
scaraAfastG4Est1,   scaraLevanPc4EM,   scaraAproxG4EM,   pc4EM,   pc4Montada,
scaraAfastG4EM, scaraAprox4Centro4        :Lugar;


est1, est2, EM, scara, xr4                        :Lugar(1);

t1_01, t1_02, t1_03, t1_04, t1_05, t1_06, t1_07, t1_08,
t1_09, t1_10, t1_11, t1_12, t1_13, t1_14, t1_15, t1_16,
t1_17, t1_18, t1_19, t1_20, t1_21,

t2_01, t2_02, t2_03, t2_04, t2_05, t2_06, t2_07, t2_08,
t2_09, t2_10, t2_11, t2_12, t2_13,

t3_01, t3_02, t3_03, t3_04, t3_05, t3_06, t3_07, t3_08,
t3_09, t3_10, t3_11, t3_12,

t4_01, t4_02, t4_03, t4_04, t4_05, t4_06, t4_07, t4_08,
t4_09, t4_10, t4_11, t4_12                        :Transicao;

Estrutura
t1_01:(est1),(est1TranspPc1);
t1_02:(est1TranspPc1),(pc1FimEst1);
t1_03:(pc1FimEst1,scara,pc2Montada),(scaraAprox1Est1);
t1_04:(scaraAprox1Est1),(scaraAproxG1Est1);
t1_05:(scaraAproxG1Est1),(scaraSegurPc1);
t1_06:(scaraSegurPc1),(scaraAfastG1Est1);
t1_07:(scaraAfastG1Est1),(scaraLevanPc1EM,est1);
t1_08:(scaraLevanPc1EM),(scaraAproxG1EM);
t1_09:(scaraAproxG1EM),(pc1EM);
t1_10:(pc1EM),(scaraAfastG1EM);
t1_11:(scaraAfastG1EM),(scaraAprox1Centro1);
t1_12:(scaraAprox1Centro1,xr4),(xr4AproxEM,scara);
t1_13:(xr4AproxEM),(xr4AproxGEM);
t1_14:(xr4AproxGEM),(xr4SegurPeca);
t1_15:(xr4SegurPeca),(xr4AfastGEM);
t1_16:(xr4AfastGEM),(xr4LevanPcFMC3,EM);
t1_17:(xr4LevanPcFMC3),(xr4AproxGFMC3);
t1_18:(xr4AproxGFMC3),(pecaFMC3);
t1_19:(pecaFMC3),(xr4AfastGFMC3);
```

```
t1_20:(xr4AfastGFMC3),(xr4AproxCentro);
t1_21:(xr4AproxCentro),(xr4);

t2_01:(est2),(est2TranspPc2);
t2_02:(est2TranspPc2),(pc2FimEst2);
t2_03:(pc2FimEst2,scara,pc3Montada),(scaraAprox2Est2);
t2_04:(scaraAprox2Est2),(scaraAproxG2Est2);
t2_05:(scaraAproxG2Est2),(scaraSegurPc2);
t2_06:(scaraSegurPc2),(scaraAfastG2Est2);
t2_07:(scaraAfastG2Est2),(scaraLevanPc2EM,est2Retornando);
t2_08:(est2Retornando),(est2);
t2_09:(scaraLevanPc2EM),(scaraAproxG2EM);
t2_10:(scaraAproxG2EM),(pc2EM,pc2Montada);
t2_11:(pc2EM),(scaraAfastG2EM);
t2_12:(scaraAfastG2EM),(scaraAprox2Centro2);
t2_13:(scaraAprox2Centro2),(scara);

t3_01:(est1),(est1TranspPc3);
t3_02:(est1TranspPc3),(pc3FimEst1);
t3_03:(pc3FimEst1,scara,pc4Montada),(scaraAprox3Est1);
t3_04:(scaraAprox3Est1),(scaraAproxG3Est1);
t3_05:(scaraAproxG3Est1),(scaraSegurPc3);
t3_06:(scaraSegurPc3),(scaraAfastG3Est1);
t3_07:(scaraAfastG3Est1),(scaraLevanPc3EM,est1);
t3_08:(scaraLevanPc3EM),(scaraAproxG3EM);
t3_09:(scaraAproxG3EM),(pc3EM,pc3Montada);
t3_10:(pc3EM),(scaraAfastG3EM);
t3_11:(scaraAfastG3EM),(scaraAprox3Centro3);
t3_12:(scaraAprox3Centro3),(scara);

t4_01:(est1),(est1TranspPc4);
t4_02:(est1TranspPc4),(pc4FimEst1);
t4_03:(pc4FimEst1,scara),(scaraAprox4Est1);
t4_04:(scaraAprox4Est1),(scaraAproxG4Est1);
t4_05:(scaraAproxG4Est1),(scaraSegurPc4);
t4_06:(scaraSegurPc4),(scaraAfastG4Est1);
t4_07:(scaraAfastG4Est1,EM),(scaraLevanPc4EM,est1);
t4_08:(scaraLevanPc4EM),(scaraAproxG4EM);
t4_09:(scaraAproxG4EM),(pc4EM,pc4Montada);
t4_10:(pc4EM),(scaraAfastG4EM);
t4_11:(scaraAfastG4EM),(scaraAprox4Centro4);
t4_12:(scaraAprox4Centro4),(scara);

Fim
Variaveis

pv_est1                                                    :Interna(4);

input1s, input2s, input3s, input4s,
input5s, input6s, input7s, input8s,
output1s, output2s, output3s, output4s,
output5s, output6s, output7s, output8s,

input1x, input2x, input3x, input4x,
input5x, input6x, input7x, input8x,
output1x, output2x, output3x, output4x,
output5x, output6x, output7x, output8x,

pms, pmx, es, ex,        {pms/r : prox. mov. scara/xr4}
gripperS, egS,
```

```
alteraASs, alteraVSs,
alteraASx, alteraVSx,
np1, np2, np3, np4,
npr1, npr2, npr3, npr4,
aux1s, aux2s, conta, contador                    :Externa;

Etiquetas
t1_01:(pv_est1=1,input2s=0,input6s=1),(pv_est1=4,aux1s=40);
t1_02:(input2s=1),(output6s=1, aux1s=0);
t1_03:(),(pms=2);
t1_04:(es=0),(pms=3);
t1_05:(es=0),(pms=4);
t1_06:(es=0),(pms=5);

t1_07:(es=0),(alteraASs=60,alteraVSs=60,pms=6,output6s=0);
t1_08:(es=0),(alteraASs=85,alteraVSs=85,pms=7);
t1_09:(es=0),(pms=8);
t1_10:(es=0),(pms=9);
t1_11:(es=0),(pms=10);

t1_12:(es=0),(pmx=2);
t1_13:(ex=0),(pmx=3);
t1_14:(ex=0),(pmx=4);
t1_15:(ex=0),(pmx=5);
t1_16:(ex=0),(pmx=6);
t1_17:(ex=0),(alteraASx=60,alteraVSx=60,pmx=7);
t1_18:(ex=0),(alteraASx=85,alteraVSx=85,pmx=8);
t1_19:(ex=0),(pmx=9);
t1_20:(ex=0),(pmx=10);
t1_21:(ex=0),();

t2_01:(input7s=1,input1s=0),(aux2s= 40);
t2_02:(input1s=1),(aux2s=0);
t2_03:(),(pms=12);
t2_04:(es=0),(pms=13);
t2_05:(es=0),(pms=14);
t2_06:(es=0),(pms=15);
t2_07:(es=0),(pms=16,aux2s= -40, contador=0, conta=1, output7s=1);

t2_08:(contador>17),(aux2s=0, conta=0);
t2_09:(es=0),(alteraASs=60,alteraVSs=60,pms=17);
t2_10:(es=0),(alteraASs=85,alteraVSs=85,pms=18,output7s=0);
t2_11:(es=0),(pms=19);
t2_12:(es=0),(pms=20);
t2_13:(es=0),();

t3_01:(pv_est1=3,input2s=0,input6s=1),(pv_est1=1,aux1s=40);
t3_02:(input2s=1),(output6s=1, aux1s=0);
t3_03:(),(pms=22);
t3_04:(es=0),(pms=23);
t3_05:(es=0),(pms=24);
t3_06:(es=0),(pms=25);
t3_07:(es=0),(pms=26,output6s=0);
t3_08:(es=0),(alteraASs=60,alteraVSs=60,pms=27);
t3_09:(es=0),(alteraASs=85,alteraVSs=85,pms=28);
t3_10:(es=0),(pms=29);
t3_11:(es=0),(pms=30);
t3_12:(es=0),();

t4_01:(pv_est1=4,input2s=0,input6s=1),(pv_est1=3,aux1s=40);
```

```
t4_02:(input2s=1),(output6s=1, aux1s=0);
t4_03:(),(pms=32);
t4_04:(es=0),(pms=33);
t4_05:(es=0),(pms=34);
t4_06:(es=0),(pms=35);
t4_07:(es=0),(pms=36,output6s=0);
t4_08:(es=0),(alteraASs=60,alteraVSs=60,pms=37);
t4_09:(es=0),(alteraASs=85,alteraVSs=85,pms=38);
t4_10:(es=0),(pms=39);
t4_11:(es=0),(pms=40);
t4_12:(es=0),();

Fim_Rede.
```

ANEXO IV: VARIÁVEIS EXTERNAS

Variáveis Externas da FMC1:

entradas do controlador do XR4:
input1, input2, input3, input4,
input5, input6, input7, input8,

saídas do controlador do XR4:
output1, output2, output3, output4,
output5, output6, output7, output8,

número total de peças boas:
npb1, npb2, npb3, npb4,

número total de peças ruins:
npr1, npr2, npr3, npr4,

posição destino para o próximo movimento do robô:
pmr,

estado da garra (0: aberta, 1: fechada):
eg,

para abrir/fechar (1/-1) a garra:
gripper,

estado do robô (0:parado e 1:movendo-se)
er,

sinal para usinagem das peças i:
usina1, usina2, usina3, usina4,

sinal para inspeção das peças i:
inspeciona1, inspeciona2, inspeciona3, inspeciona4,

probabilidade da peça i:
probPeca1, probPeca2, probPeca3, probPeca4,

para alterar a velocidade e a aceleração do sistema:
alteraAS, alteraVS.

Variáveis Externas da FMC2:

entradas dos controladores do XR4('x') e do SCARA('s'):

```
input1s, input2s, input3s, input4s,
input5s, input6s, input7s, input8s,
input1x, input2x, input3x, input4x,
input5x, input6x, input7x, input8x,
```

saídas dos controladores do XR4 e do SCARA:

```
output1s, output2s, output3s, output4s,
output5s, output6s, output7s, output8s,
output1x, output2x, output3x, output4x,
output5x, output6x, output7x, output8x,
```

posição destino para o próximo movimento do robô:

```
pmS,pmX,
```

estado da garra:

```
egS,egX,
```

para abrir/fechar (1/-1) a garra:

```
gripperS,gripperX,
```

estado do robô (0:parado e 1:movendo-se)

```
eS,eX,
```

sinal para gerar um numero aleatorio:

```
gAleat,
```

para alterar a velocidade e a aceleração do sistema:

```
alteraASs, alteraVSs,
alteraASx, alteraVSx,
```

número aleatório gerado:

```
na1, na2, na3, na4,
```

atribuir o nível e a direção das portas auxiliares:

```
aux1s, aux2s,
aux1x, aux2x,
```

se 'conta'=1 então começa a contar em 'contador':

```
contador, conta.
```